Word-Level Abstractions for Sequential Design Verification using Algebraic Geometry

Xiaojun Sun, PhD Candidate

Electrical and Computer Engineering, University of Utah

Advisor: *Priyank Kalla*

Dec 16, 2016

Outline

- Contributions
- Motivations
- Previous Work
- Preliminaries
  - Finite fields
  - Polynomial algebra & Algebraic geometry
  - Projection of varieties
- Projection based abstraction
  - Application: Reachability analysis
  - Application: Sequential arithmetic ckt verification
- UNSAT core based abstraction
  - UNSAT core extraction using Gröbner basis refutation
  - Application: Bounded model checking (BMC) with abstraction refinement
- Conclusion & Future work

Contributions

- Word-level reachability analysis – analog of implicit state enumeration
    - **Q:** Why word-level?

Contributions

- Word-level reachability analysis – analog of implicit state enumeration
    - **Q:** Why word-level?
    - **A:** Data $\leftarrow$ word-level info

Contributions

- Word-level reachability analysis – analog of implicit state enumeration
  - **Q:** Why word-level?
  - **A:** Data $\leftarrow$ word-level info
  - **A:** Simplify representation (abstraction) of state-space $\rightarrow$ Efficiency!

Contributions

- Word-level reachability analysis – analog of implicit state enumeration
  - **Q:** Why word-level?
  - **A:** Data $\leftarrow$ word-level info
  - **A:** Simplify representation (abstraction) of state-space $\rightarrow$ Efficiency!
- Apply word-level reachability algorithm to sequential arithmetic circuit verification
  - Abstraction $\rightarrow$ word-level signature each time-frame
    - Word-level abstraction from bit-level ckts [Pruss, 2015]
  - Word-level unrolling

Contributions

- Word-level reachability analysis – analog of implicit state enumeration
  - **Q:** Why word-level?
  - **A:** Data $\leftarrow$ word-level info
  - **A:** Simplify representation (abstraction) of state-space $\rightarrow$ Efficiency!
- Apply word-level reachability algorithm to sequential arithmetic circuit verification
  - Abstraction $\rightarrow$ word-level signature each time-frame
    - Word-level abstraction from bit-level ckts [Pruss, 2015]
  - Word-level unrolling
- UNSAT cores in Algebraic geometry
  - Analyze Buchberger's algorithm $\rightarrow$ extract refutation proof $\rightarrow$ UNSAT core
  - Structure of refutation proof $\rightarrow$ refine UNSAT core

Contributions

- Word-level reachability analysis – analog of implicit state enumeration
  - **Q:** Why word-level?
  - **A:** Data $\leftarrow$ word-level info
  - **A:** Simplify representation (abstraction) of state-space $\rightarrow$ Efficiency!
- Apply word-level reachability algorithm to sequential arithmetic circuit verification
  - Abstraction $\rightarrow$ word-level signature each time-frame
    - Word-level abstraction from bit-level ckts [Pruss, 2015]
  - Word-level unrolling
- UNSAT cores in Algebraic geometry
  - Analyze Buchberger's algorithm $\rightarrow$ extract refutation proof $\rightarrow$ UNSAT core
  - Structure of refutation proof $\rightarrow$ refine UNSAT core
- Implement above algos: C++ & SINGULAR
  - For sequential GF multipliers: overwhelmingly better than contemporary tools

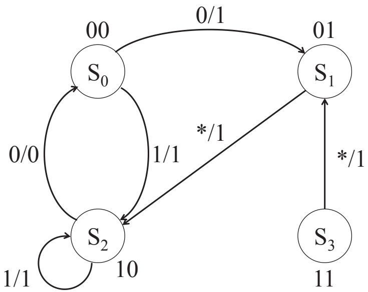


Figure : State Transition Graph

- Initial state: $\{00\}$
- Iteration 1:
  - Start from $\{00\}$
  - One-step transition: $\{01, 10\}$
  - Newly reached: $\{01, 10\}$
- Iteration 2:
  - Start from $\{01, 10\}$
  - One-step transition: $\{00, 10\}$
  - Newly reached: $\emptyset$
- All reachable states detected. Final reached states: $\{00, 01, 10\}$

Motivation I: BFS state space traversal

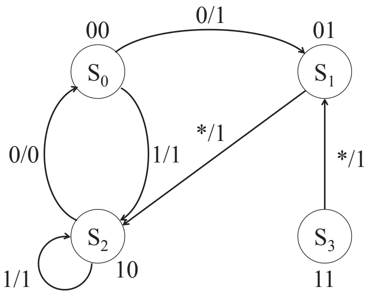
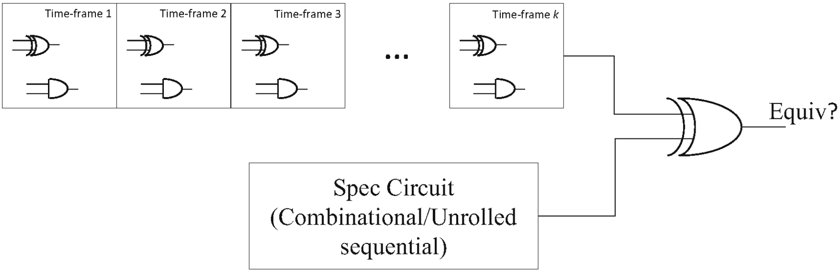


Figure : State Transition Graph

- Initial state: $\{00\}$
- Iteration 1:
  - Start from $\{00\}$
  - One-step transition: $\{01, 10\}$
  - Newly reached: $\{01, 10\}$
- Iteration 2:
  - Start from $\{01, 10\}$
  - One-step transition: $\{00, 10\}$
  - Newly reached: $\emptyset$
- All reachable states detected. Final reached states: $\{00, 01, 10\}$
- Still need bit-level Boolean variables to represent states

Go back to example

Motivation II: Sequential arithmetic circuits verification



- Problem: Verify the function of sequential arithmetic circuit
  - Operands preloaded into registers
  - After k clock cycles, give desired output
- Conventional: explicitly unroll k time-frames (bit-level) and setup miter
  - Check miter output: SAT, BDDs, AIGs
  - Bit-blasting!

Go back to Moore machine

Motivation III: k-BMC with abstraction refinement

ALGORITHM: k-BMC with Abstraction Refinement (L. Zhang'05)

Input: M is the original machine, p is the property to check, k is the number of steps unrolling M

```
1  k = InitValue;
2  if k-BMC(M, p, k) is SAT (reachable states after k steps violates p) then
3      return "Found error trace"
4  else
5      Extract UNSAT core 𝒫 of k-BMC ;
6      M′ = ABSTRACT(M, 𝒫);
7  end
8  if MODEL-CHECK(M′, p) returns PASS then
9      return "Passing property"
10 else
11     Increase bound k;
12     goto Line 2;
13 end
```

Go back to Abs refine

Previous work

- Sequential Equivalence Checking (SEC): bit-blasting, or structural info dependency
  - Usually based on reachability analysis
  - Sequential miter
  - Unroll, then use DDs, SAT or AIGs (Combinational)
  - Induction-based
- Symbolic model checking (counterexample, IC3): SAT/BDDs in nature
- Word-level techniques (term rewriting, uninterpreted function): ?/no encoding
- Algebraic geometry methods
  - Gröber basis in model checking [Avrunin,CAV'96;Vardi,IASTED'07]: Analog of bit-level Boolean functions
  - Abstract word-level polynomial representation for arbitrary combinational ckt [Pruss,TCAD'16]

Previous work

- Sequential Equivalence Checking (SEC): bit-blasting, or structural info dependency
  - Usually based on reachability analysis
  - Sequential miter
  - Unroll, then use DDs, SAT or AIGs (Combinational)
  - Induction-based
- Symbolic model checking (counterexample, IC3): SAT/BDDs in nature
- Word-level techniques (term rewriting, uninterpreted function): ?/no encoding
- Algebraic geometry methods
  - Gröber basis in model checking [Avrunin,CAV'96;Vardi,IASTED'07]: Analog of bit-level Boolean functions
  - Abstract word-level polynomial representation for arbitrary combinational ckt [Pruss,TCAD'16]
- No purely word-level sequential verification: data/abstraction/algorithm

Working field: $\mathbb{F}_{2^k}$

- Our proposed state-space model is based on finite field $\mathbb{F}_{2^k}$
  - $\mathbb{B}^k$: bit-vector
  - $\mathbb{Z}_{2^k}$, $\mathbb{R}_q$: approaches not compatible
- Evaluations in $\mathbb{B}^k \Leftrightarrow$ Elements in $\mathbb{F}_{2^k}$
- Functions $\mathbb{B}^k \rightarrow \mathbb{B}^k \Leftrightarrow \mathcal{F} : \mathbb{F}_{2^k} \rightarrow \mathbb{F}_{2^k}$

Galois field $\mathbb{F}_q$ is a finite field with q elements, $q = p^k$, $p =$prime

- 0,1 elements, commutative, associate, distributive laws
- Closure property: $+, -, \times$, inverse $(\div)$

Our interest: $\mathbb{F}_q = \mathbb{F}_{2^k}$ $(q = 2^k)$

- $\mathbb{F}_{2^k}$: k-dimensional extension of $\mathbb{F}_2 = \{0, 1\}$
  - k-bit bit-vector, AND/XOR arithmetic

To construct $\mathbb{F}_{2^k}$

- $\mathbb{F}_{2^k} \equiv \mathbb{F}_2[x] \pmod{P(x)}$
- $P(x) \in \mathbb{F}_2[x]$, irreducible polynomial of degree k
- Operations performed (mod $P(x)$) and coefficient reduced (mod 2)
  - E.g. $\mathbb{C} = \mathbb{R}[x] \pmod{x^2 + 1}$

Consider: $\mathbb{F}_{2^3} = \mathbb{F}_2[x] \pmod{x^3+x+1}$
$A \in \mathbb{F}_2[x]$
$A \pmod{x^3+x+1} = a_2x^2 + a_1x + a_0$. Let $P(\alpha) = 0$:

- $\langle a_2, a_1, a_0 \rangle = \langle 0,0,0 \rangle = 0$
- $\langle a_2, a_1, a_0 \rangle = \langle 0,0,1 \rangle = 1$
- $\langle a_2, a_1, a_0 \rangle = \langle 0,1,0 \rangle = \alpha$
- $\langle a_2, a_1, a_0 \rangle = \langle 0,1,1 \rangle = \alpha + 1$
- $\langle a_2, a_1, a_0 \rangle = \langle 1,0,0 \rangle = \alpha^2$
- $\langle a_2, a_1, a_0 \rangle = \langle 1,0,1 \rangle = \alpha^2 + 1$
- $\langle a_2, a_1, a_0 \rangle = \langle 1,1,0 \rangle = \alpha^2 + \alpha$
- $\langle a_2, a_1, a_0 \rangle = \langle 1,1,1 \rangle = \alpha^2 + \alpha + 1$

Theorem (Fermat's Little Theorem over $\mathbb{F}_q$)

Let $\alpha \in \mathbb{F}_q$, then $\alpha^q = \alpha$. Therefore, $x^q - x$ vanishes on all points in $\mathbb{F}_q$.

$\{a_2 a_1 a_0\} \in \mathbb{B}^3$	$A \in \mathbb{F}_{2^3}$	$\to$	$\{z_2 z_1 z_0\} \in \mathbb{B}^3$	$Z \in \mathbb{F}_{2^3}$
000	0	$\to$	000	0
001	1	$\to$	001	1
010	α	$\to$	111	$\alpha^2 + \alpha + 1$
011	$\alpha + 1$	$\to$	111	$\alpha^2 + \alpha + 1$
100	α^2	$\to$	101	$\alpha^2 + 1$
101	$\alpha^2 + 1$	$\to$	011	$\alpha + 1$
110	$\alpha^2 + \alpha$	$\to$	101	$\alpha^2 + 1$
111	$\alpha^2 + \alpha + 1$	$\to$	101	$\alpha^2 + 1$

Table : Truth table for mappings in $\mathbb{B}^3$ and $\mathbb{F}_{2^3}$

Theorem (Fermat's Little Theorem over $\mathbb{F}_q$)

Let $\alpha \in \mathbb{F}_q$, then $\alpha^q = \alpha$. Therefore, $x^q - x$ vanishes on all points in $\mathbb{F}_q$.

$\{a_2 a_1 a_0\} \in \mathbb{B}^3$	$A \in \mathbb{F}_{2^3}$	$\to$	$\{z_2 z_1 z_0\} \in \mathbb{B}^3$	$Z \in \mathbb{F}_{2^3}$
000	0	$\to$	000	0
001	1	$\to$	001	1
010	α	$\to$	111	$\alpha^2 + \alpha + 1$
011	$\alpha + 1$	$\to$	111	$\alpha^2 + \alpha + 1$
100	α^2	$\to$	101	$\alpha^2 + 1$
101	$\alpha^2 + 1$	$\to$	011	$\alpha + 1$
110	$\alpha^2 + \alpha$	$\to$	101	$\alpha^2 + 1$
111	$\alpha^2 + \alpha + 1$	$\to$	101	$\alpha^2 + 1$

Table : Truth table for mappings in $\mathbb{B}^3$ and $\mathbb{F}_{2^3}$

$$
\begin{aligned}
Z =& \mathcal{F}(A) \\
=& (\alpha^2 + \alpha + 1)A^7 + (\alpha^2 + 1)A^6 + \alpha A^5 + (\alpha + 1)A^4 \\
&+ (\alpha^2 + \alpha + 1)A^3 + (\alpha^2 + 1)A
\end{aligned}
$$

Preliminaries: Computer algebra terminology

Let $\mathbb{F}_q = GF(2^k)$, and $\overline{\mathbb{F}_q}$ be its closure

- $\mathbb{F}_q[x_1, \ldots, x_n]$: ring of all polynomials with coefficients in $\mathbb{F}_q$
- Polynomial $f = c_1X_1 + c_2X_2 + \cdots + c_tX_t$
  - A monomial ordering is imposed on $f : X_1 > X_2 > \cdots > X_t$
  - Leading term $lt(f) = c_1X_1, \quad tail(f) = c_2X_2 + \cdots + c_tX_t$
  - Leading coefficient $lt(f) = c_1$ and leading monomial $lm(f) = X_1$
  - LEX $x > y > z$: $\quad f = -2x^3 + 2x^2yz + 3xy^3$
  - DEGLEX $x > y > z$: $\quad f = 2x^2yz + 3xy^3 - 2x^3$
  - DEGREVLEX $x > y > z$: $\quad f = 3xy^3 + 2x^2yz - 2x^3$
- Leading terms $lt(f)$ play an important role
  - Affect division results!

Preliminaries: Polynomial division

Divide $f = x^3 - 2x^2 + 2x + 8$ by $g = 2x^2 + 3x + 1$

Preliminaries: Polynomial division

Divide $f = x^3 - 2x^2 + 2x + 8$ by $g = 2x^2 + 3x + 1$

$$
\begin{array}{r|rrrr}
 & & & \frac{1}{2}x & -\frac{7}{4} \\
\hline
2x^2 + 3x + 1 & x^3 & -2x^2 & +2x & +8 \\
 & -x^3 & -\frac{3}{2}x^2 & -\frac{1}{2}x & \\
\cline{2-4}
 & & -\frac{7}{2}x^2 & +\frac{3}{2}x & +8 \\
 & & \frac{7}{2}x^2 & +\frac{21}{4}x & +\frac{7}{4} \\
\cline{3-5}
 & & & \frac{27}{4}x & +\frac{39}{4}
\end{array}
$$

Divide $f = x^3 - 2x^2 + 2x + 8$ by $g = 2x^2 + 3x + 1$

$$
\begin{array}{r|rrrr}
 & & & \frac{1}{2}x & -\frac{7}{4} \\
\hline
2x^2+3x+1 & x^3 & -2x^2 & +2x & +8 \\
 & -x^3 & -\frac{3}{2}x^2 & -\frac{1}{2}x & \\
\cline{2-4}
 & & -\frac{7}{2}x^2 & +\frac{3}{2}x & +8 \\
 & & \frac{7}{2}x^2 & +\frac{21}{4}x & +\frac{7}{4} \\
\cline{3-5}
 & & & \frac{27}{4}x & +\frac{39}{4}
\end{array}
$$

- The key step in division: $r = f - \frac{lt(f)}{lt(g)} \cdot g$, denoted $f \xrightarrow{g} r$
- Similarly divide f by a set of polynomials $F = \{f_1, \ldots, f_s\}$
- Denoted: $f \xrightarrow{f_1,\ldots,f_s}_+ r$
  - Remainder r is reduced: no term in r is divisible by $lt(f_i)$

Let $\mathbb{F}_q = GF(2^k)$:

- Given a set of polynomials:
  - $f_1, f_2, \ldots, f_s \in \mathbb{F}_q[x_1, \ldots, x_n]$
  - Find solutions to $f_1 = f_2 = \cdots = f_s = 0$
- Variety: Set of ALL solutions to a given system of polynomial equations: $V(f_1, \ldots, f_s)$
  - In $\mathbb{R}[x, y]$, $V(x^2 + y^2 - 1) = \{all\ points\ on\ circle : x^2 + y^2 - 1 = 0\}$
  - In $\mathbb{R}[x]$, $V(x^2 + 1) = \emptyset$
  - In $\mathbb{C}[x]$, $V(x^2 + 1) = \{(\pm i)\}$
- Variety depends on the ideal generated by the polynomials.
- Reason about the Variety by analyzing the Ideals

Definition

Ideals of Polynomials: Let $f_1, f_2, \ldots, f_s \in \mathbb{F}_q[x_1, \ldots, x_n]$. Let

$$J = \langle f_1, f_2 \ldots, f_s \rangle = \{f_1 h_1 + f_2 h_2 + \cdots + f_s h_s\}, \quad h_i \in \mathbb{F}_q[x_1, \ldots, x_n]$$

$J = \langle f_1, f_2 \ldots, f_s \rangle$ is an ideal generated by $f_1, \ldots, f_s$ and the polynomials are called the generators.

- Different generators can generate the same ideal
- $\langle f_1, \cdots, f_s \rangle = \cdots = \langle g_1, \cdots, g_t \rangle$
- Some generators are a “better” representation of the ideal
- A **Gröbner basis** G is a “canonical” representation of an ideal
  - $I = \langle F \rangle = \langle G \rangle$, and $V(F) = V(G)$
- Map: set of states $\rightarrow$ variety of polynomial ideal

INPUT : $F = \{f_1, \ldots, f_s\}$
OUTPUT : $G = \{g_1, \ldots, g_t\}$
$G := F$;
REPEAT
  $G' := G$
  For each pair $\{f, g\}, f \neq g$ in G' DO
    $Spoly(f, g) \xrightarrow{G'}_+ r$
    IF $r \neq 0$ THEN $G := G \cup \{r\}$
UNTIL $G = G'$

INPUT : $F = \{f_1, \ldots, f_s\}$
OUTPUT : $G = \{g_1, \ldots, g_t\}$
$G := F$;
REPEAT
  $G' := G$
  For each pair $\{f, g\}, f \neq g$ in G' DO
    $Spoly(f, g) \xrightarrow{G'}_{+} r$
    IF $r \neq 0$ THEN $G := G \cup \{r\}$
UNTIL $G = G'$

- $Spoly(f, g) = \frac{L}{lt(f)} \cdot f - \frac{L}{lt(g)} \cdot g$
  $L = \mathrm{LCM}(lm(f), lm(g))$, $lm(f)$: leading monomial of f
- Animation 1...

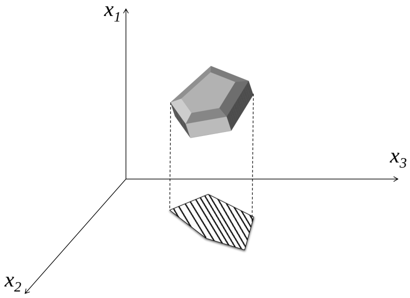
INPUT : $F = \{f_1, \ldots, f_s\}$
OUTPUT : $G = \{g_1, \ldots, g_t\}$
$G := F$;
REPEAT
  $G' := G$
  For each pair $\{f, g\}, f \neq g$ in G' DO
    $Spoly(f, g) \xrightarrow{G'}_{+} r$
    IF $r \neq 0$ THEN $G := G \cup \{r\}$
UNTIL $G = G'$

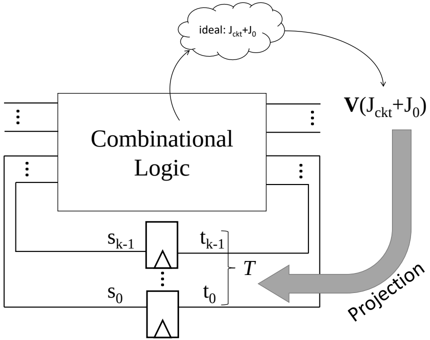
- $Spoly(f, g) = \frac{L}{lt(f)} \cdot f - \frac{L}{lt(g)} \cdot g$
  $L = \text{LCM}(lm(f), lm(g))$, $lm(f)$: leading monomial of f
- Animation 1...
- GB enables mapping: set of states $\rightarrow$ variety of polynomial ideal
  - Algebraic/reasoning engine: application to elimination
- GB enables abstraction

Gróbner basis with elimination term order

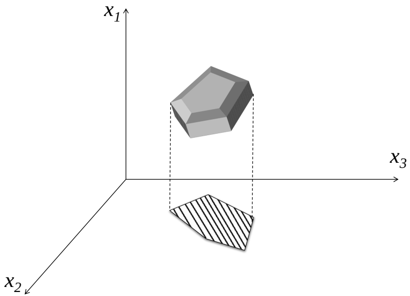
- Let ideal $I = \langle f_1, f_2, f_3 \rangle$ where
  - $f_1 = x^2 + y + z - 1$
  - $f_2 = x + y^2 + z - 1$
  - $f_3 = x + y + z^2 - 1$
- The Gröbner basis of I with elimination (LEX) order ($x > y > z$) is
  - $g_1 = x + y + z^2 - 1$
  - $g_2 = y^2 - y - z^2 + z$
  - $g_3 = 2yz^2 + z^4 - z^2$
  - $g_4 = z^6 - 4z^4 + 4z^3 - z^2$
- Notice that g_2 and g_3 only contain variables y and z
  - Eliminates variable $x \Leftrightarrow \exists_x$ in Boolean formula!
- Similarly, g_4 only contains the variable z and eliminates x and y

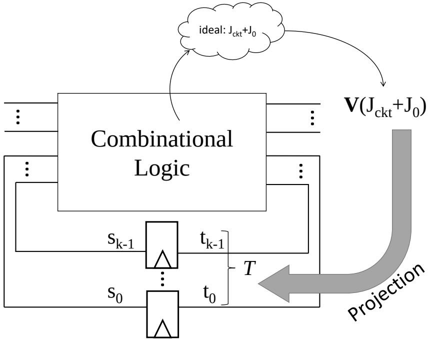
- Let ideal $I = \langle f_1, f_2, f_3 \rangle$ where
  - $f_1 = x^2 + y + z - 1$
  - $f_2 = x + y^2 + z - 1$
  - $f_3 = x + y + z^2 - 1$
- The Gröbner basis of I with elimination (LEX) order $(x > y > z)$ is
  - $g_1 = x + y + z^2 - 1$
  - $g_2 = y^2 - y - z^2 + z$
  - $g_3 = 2yz^2 + z^4 - z^2$
  - $g_4 = z^6 - 4z^4 + 4z^3 - z^2$
- Notice that g_2 and g_3 only contain variables y and z
  - Eliminates variable $x \Leftrightarrow \exists_x$ in Boolean formula!
- Similarly, g_4 only contains the variable z and eliminates x and y
- $\mathrm{GB}(I_{x,y,z}) \cap \mathbb{F}_q[z]$ related to projection on variable z!





- Projection of variety from $\mathbb{F}[x_1, x_2, x_3]$ to $\mathbb{F}[x_2, x_3]$
- Projection of ckt ideal's variety on next state (NS) variables T





- Projection of variety from $\mathbb{F}[x_1, x_2, x_3]$ to $\mathbb{F}[x_2, x_3]$
- Projection of ckt ideal's variety on next state (NS) variables T
- $\text{GB}(J_{ckt} + J_0) \cap \mathbb{F}_2[T] \implies$ Next state polynomial $f(T)$!

BFS traversal algorithm

ALGORITHM: Breadth-first Traversal Algorithm

Input: Transition functions Δ, initial state S^0

$from^0 = reached = S^0$;
repeat
    $i \leftarrow i + 1$;
    $to^i \leftarrow \mathrm{Img}(\Delta, from^{i-1})$;
    $new^i \leftarrow to^i \cap \overline{reached}$;
    $reached \leftarrow reached \cup new^i$;
    $from^i \leftarrow new^i$;
until $new^i == 0$;
return *reached*

BFS traversal algorithm

ALGORITHM: Breadth-first Traversal Algorithm

Input: Transition functions Δ, initial state S^0

$from^0 = reached = S^0$;
repeat
    $i \leftarrow i + 1$;
    $to^i \leftarrow \mathrm{Img}(\Delta, from^{i-1})$;
    $new^i \leftarrow to^i \cap \overline{reached}$;
    $reached \leftarrow reached \cup new^i$;
    $from^i \leftarrow new^i$;
until $new^i == 0$;
return *reached*

- Image function:
  $\mathrm{Img}(\Delta, from) = \exists_s \exists_x [T(s,x,t) \wedge from] = \exists_s \exists_x \bigwedge_{i=1}^{n} (t_i \overline{\oplus} \Delta_i) \wedge from$
- In $\mathbb{B}^k$, image function $\Leftrightarrow \exists$
- In $\mathbb{F}_{2^k}$, need to implement quantifier elimination

Recall: Breadth-First Traversal Algorithm

ALGORITHM: Breadth-first Traversal Algorithm

Input: Transition functions Δ, initial state S^0

$from^0 = reached = S^0$;
repeat
  $i \leftarrow i + 1$;
  $to^i \leftarrow \mathrm{Img}(\Delta, from^{i-1})$;
  $new^i \leftarrow to^i \cap \overline{reached}$;
  $reached \leftarrow reached \cup new^i$;
  $from^i \leftarrow new^i$;
until $new^i == 0$;
return *reached*

Implement this algorithm by finding analogs in algebraic geometry

Recall: Breadth-First Traversal Algorithm

ALGORITHM: Breadth-first Traversal Algorithm

Input: Transition functions Δ, initial state S^0; // polynomial ideal

$from^0 = reached = S^0$; // set of states $\Leftrightarrow$ variety of ideal

repeat

$\quad i \leftarrow i + 1$;

$\quad to^i \leftarrow \mathrm{Img}(\Delta, from^{i-1})$;

$\quad new^i \leftarrow to^i \cap \overline{reached}$;

$\quad reached \leftarrow reached \cup new^i$;

$\quad from^i \leftarrow new^i$;

until $new^i == 0$;

return *reached*

Implement this algorithm by finding analogs in algebraic geometry

Recall: Breadth-First Traversal Algorithm

ALGORITHM: Breadth-first Traversal Algorithm

Input: Transition functions Δ, initial state S^0; // polynomial ideal

$from^0 = reached = S^0$; // set of states $\Leftrightarrow$ variety of ideal

repeat

  $i \leftarrow i + 1$;

  $to^i \leftarrow \text{Img}(\Delta, from^{i-1})$; // GB of Elim ideal

  $new^i \leftarrow to^i \cap \overline{reached}$;

  $reached \leftarrow reached \cup new^i$;

  $from^i \leftarrow new^i$;

until $new^i == 0$;

return $reached$

Implement this algorithm by finding analogs in algebraic geometry

Recall: Breadth-First Traversal Algorithm

ALGORITHM: Breadth-first Traversal Algorithm

Input: Transition functions Δ, initial state S^0; // polynomial ideal

$\mathit{from}^0 = \mathit{reached} = S^0$; // set of states $\Leftrightarrow$ variety of ideal

repeat

    $i \leftarrow i + 1$;

    $\mathit{to}^i \leftarrow \mathrm{Img}(\Delta, \mathit{from}^{i-1})$; // GB of Elim ideal

    $\mathit{new}^i \leftarrow \mathit{to}^i \cap \overline{\mathit{reached}}$; // ideal quotient & sum

    $\mathit{reached} \leftarrow \mathit{reached} \cup \mathit{new}^i$;

    $\mathit{from}^i \leftarrow \mathit{new}^i$;

until $\mathit{new}^i == 0$;

return *reached*

Implement this algorithm by finding analogs in algebraic geometry

Recall: Breadth-First Traversal Algorithm

ALGORITHM: Breadth-first Traversal Algorithm

Input: Transition functions Δ, initial state S^0; // polynomial ideal

$from^0 = reached = S^0$; // set of states $\Leftrightarrow$ variety of ideal

repeat

  $i \leftarrow i + 1$;

  $to^i \leftarrow \mathrm{Img}(\Delta, from^{i-1})$; // GB of Elim ideal

  $new^i \leftarrow to^i \cap \overline{reached}$; // ideal quotient & sum

  $reached \leftarrow reached \cup new^i$; // ideal product

  $from^i \leftarrow new^i$;

until $new^i == 0$;

return *reached*

Implement this algorithm by finding analogs in algebraic geometry

Intersection and union in algebraic geometry

Definition

(**Sum/Product of Ideals**) If $I = \langle f_1, \ldots, f_r \rangle$ and $J = \langle g_1, \ldots, g_s \rangle$ are ideals in $\mathbb{F}[x_1, \ldots, x_n]$, then the **sum** of I and J is defined as

$$I + J = \langle f_1, \ldots, f_r, g_1, \ldots, g_s \rangle$$

And the **product** of I and J is defined as

$$I \cdot J = \langle f_i g_j \mid 1 \leq i \leq r, 1 \leq j \leq s \rangle$$

Theorem

If I and J are ideals in $\mathbb{F}[x_1, \ldots, x_n]$, then $\mathbf{V}(I + J) = \mathbf{V}(I) \bigcap \mathbf{V}(J)$ and $\mathbf{V}(I \cdot J) = \mathbf{V}(I) \bigcup \mathbf{V}(J)$.

Definition

(**Quotient of Ideals**) If I and J are ideals in $\mathbb{F}[x_1, \ldots, x_n]$, then $I : J$ is the set

$$\{f \in \mathbb{F}[x_1, \ldots, x_n] \mid f \cdot g \in I, \forall g \in J\}$$

and is called the **ideal quotient** of I by J.

Theorem

Let J_0 be an ideal of vanishing polynomials over $\mathbb{F}_{2^k}[x_1, \ldots, x_n]$, then

$$\mathbf{V}(J_0 : J) = \mathbf{V}(J_0) - \mathbf{V}(J) = \overline{\mathbf{V}(J)}$$

- $V(J) \subseteq \mathbb{F}_{2^k}$ in affine space
- Given ideal J, compute J' s.t. $V(J') = \overline{V(J)} = \mathbb{F}_{2^k} - V(J) \implies J' = J_0 : J$

Our proposed algorithm of BFS traversal based on algebraic geometry

ALGORITHM: Algebraic Geometry based FSM Traversal

Input: The circuit's characteristic polynomial ideal J_{ckt}, initial state polynomial $\mathbb{F}(S)$, and LEX term order: bit-level variables $x, s, t >$ PS word $S >$ NS word T

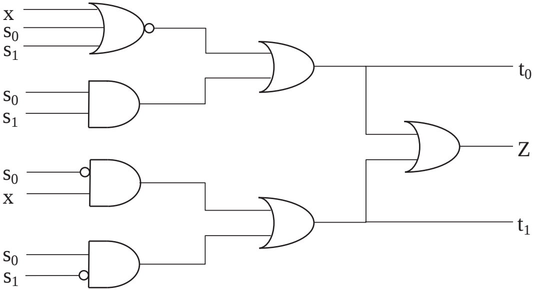
$from^0 = reached = \mathbb{F}(S)$;
repeat
  $i \leftarrow i + 1$;
  $G \leftarrow$GB$(\langle J_{ckt}, J_0, from^{i-1}\rangle)$; `// This step contains bit-level`
  $\langle to^i \rangle \leftarrow G \cap \mathbb{F}_{2^k}[T]$; `// Only word-level` S, T `onwards`
  $\langle new^i \rangle \leftarrow \langle to^i \rangle + (\langle T^{2^k} - T \rangle : \langle reached \rangle)$;
  $\langle reached \rangle \leftarrow \langle reached \rangle \cdot \langle new^i \rangle$;
  $from^i \leftarrow new^i(S \setminus T)$;
until $\langle new^i \rangle == \langle 1 \rangle$;
return $\langle reached \rangle$

Go to example page 2

- Initial state $from^0 = S(\{00\})$
- **Iteration 1:** Compose an elimination ideal J

$f_1 : t_0 - (xs_0s_1 + xs_0$
$+ xs_1 + x + s_0 + s_1 + 1)$
$f_2 : t_1 - (xs_0 + x + s_0s_1 + s_0)$
$f_3 : S - s_0 - s_1\alpha$
$f_4 : T - t_0 - t_1\alpha$

$f_5 : x^2 - x$
$f_6 : s_0^2 - s_0, f_7 : s_1^2 - s_1$
$f_8 : t_0^2 - t_0, f_9 : t_1^2 - t_1$
$f_{10} : S^4 - S, f_{11} : T^4 - T$



$J_{ckt} = \langle f_1, f_2, f_3, f_4 \rangle$
$J_0 = \langle f_5, f_6, \ldots, f_{11} \rangle$

Full Blown traversal of example circuit using algebraic geometry

- Elimination term order:

$$\{x, s_0, s_1, t_0, t_1\}\ (all\ bits)\ >\ S\ (PS\ word)\ >\ T\ (NS\ word)$$

- Compute the reduced GB for $J = J_{ckt} + J_0 + \langle from^0 \rangle$
- Next state

$$to^1 = \langle T^2 + (\alpha + 1)T + \alpha \rangle$$

- Mapping to set of states

$$V(to^1) = \{1, \alpha\} \Leftrightarrow \{01, 10\}$$

- Complement of formerly reached state:

$$\langle T^4 - T \rangle : \langle T \rangle = \langle T^3 + 1 \rangle$$

- Mapping to set of states

$$V(\langle T^3 + 1 \rangle) = \{1, \alpha, 1 + \alpha\} \Leftrightarrow \{01, 10, 11\}$$

▸ STG

Full Blown traversal of example circuit using algebraic geometry

- Newly reached states:

$$\langle T^3 + 1, T^2 + (\alpha + 1)T + \alpha \rangle = \langle T^2 + (\alpha + 1)T + \alpha \rangle \ (\{01, 10\})$$

- Update current reached states

$$reach = \langle T \cdot T^2 + (\alpha + 1)T + \alpha \rangle = \langle T^3 + (\alpha + 1)T^2 + \alpha T \rangle$$

- Mapping to set of states

$$V(reached) = \{0, 1, \alpha\} \Leftrightarrow \{00, 01, 10\}$$

- Update the present states for next iteration

$$from^1 = \langle S^2 + (\alpha + 1)S + \alpha \rangle$$

STG

Full Blown traversal of example circuit using algebraic geometry

- Iteration 2:
  - Next state: $to^2 = \langle T^2 + \alpha T \rangle$ $(\{00, 10\})$
  - The complement of *reached*:

$$\langle T^4 - T \rangle : \langle T^3 + (\alpha + 1) T^2 + \alpha T \rangle = \langle T + 1 + \alpha \rangle \; (\{11\})$$

  - Newly reached state:

$$\langle T^2 + \alpha T, T + 1 + \alpha \rangle = \langle {\color{red}1} \rangle$$

- Algorithm terminates
- Return value (final reachable states):

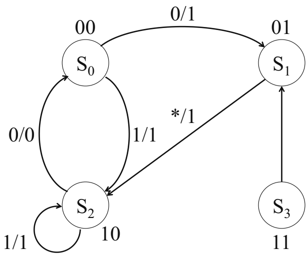
$$reached = \langle T^3 + (\alpha + 1) T^2 + \alpha T \rangle$$

Improve complexity using RATO [Pruss, '15]

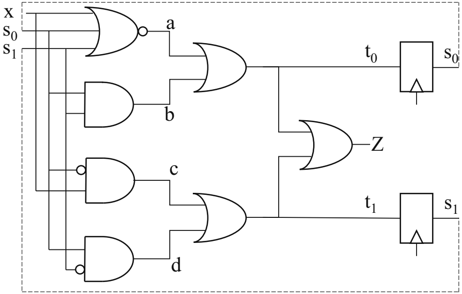
- Directly compute GB($J_{ckt} + J_0$) in $\mathbb{F}_q$ is costly ($q^{O(d)}$)
- GB sensitive to term order$\rightarrow$Transform to simpler set for GB?

Improve complexity using RATO [Pruss, '15]

- Directly compute GB($J_{ckt} + J_0$) in $\mathbb{F}_q$ is costly ($q^{O(d)}$)
- GB sensitive to term order→Transform to simpler set for GB?
- RATO: reverse topological traverse on ckt structure



(a)



(b)

Benefits of using RATO

- Definition of RATO:
  "bit-level variables ordered reverse topologically " $> T > S$
- Only one pair of poly with non-relatively-prime leading terms
  - If $gcd(lt(f_w), lt(f_g)) = 1$:
  - $Spoly(f_w, f_g) \xrightarrow{J_{ckt}+J_0}_+ 0$
- Topology in structure
  - Each poly with leading term – fanout
  - Divide with levelization
  - Only inputs (primary & pseudo) left!

Example of using RATO

- RATO: LEX with $(t_0, t_1) > (a, b, c, d) > (x, s_0, s_1) > T > S$

$$f_1 : a + xs_0s_1 + xs_0 + xs_1 + x + s_0s_1 + s_0 + s_1 + 1$$

$$f_2 : b + s_0s_1 \quad f_3 : c + x + xs_0 \quad f_4 : d + s_0s_1 + s_0$$

$$f_5 : t_0 + ab + a + 1 \quad f_6 : t_1 + cd + c + d \quad f_7 : t_0 + t_1\alpha + T$$

- *Spoly* reduction gives $T + \mathcal{F}(primary/pseudo\ inputs)$

$$Spoly(f_5, f_7) \xrightarrow{J_{ckt}+J_0}_{+} T + s_0s_1x + \alpha s_0s_1$$
$$+ (1+\alpha)s_0x + (1+\alpha)s_0 + s_1x + s_1 + (1+\alpha)x + 1$$

- **Q:** How to get rid of bit-level inputs?

Bit-to-word conversion

- Objective: find $s_i = \mathcal{G}(S)$
- Build system of poly eqn by squaring:

$$\begin{bmatrix} S \\ S^2 \\ S^{2^2} \\ \vdots \\ S^{2^{k-1}} \end{bmatrix} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{k-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(k-1)} \\ 1 & \alpha^4 & \alpha^8 & \cdots & \alpha^{4(k-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{2^{k-1}} & \alpha^{2\cdot 2^{k-1}} & \cdots & \alpha^{(k-1)\cdot 2^{k-1}} \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ \vdots \\ s_{k-1} \end{bmatrix}$$

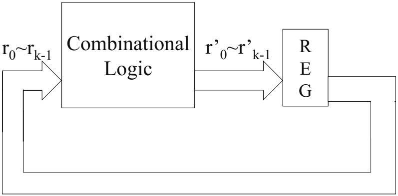
- Transition function $f_T : T + \mathbb{F}(S, x)$
- Elimination on ideal $\langle f_T, f_S \rangle + J_0'$ using $S, x > T$

Experiment results: word-level traversal

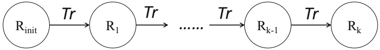
Table : Results of running benchmarks using our tool. Parts I to III denote the time taken by polynomial divisions, bit-level to word-level abstraction and iterative reachability convergence checking part of our approach, respectively.

Benchmark	# States	# iterations	Runtime (sec)			Runtime of VIS (sec)
			I	II	III	
b01	18	5	< 0.01	0.01	0.02	< 0.01
b02	8	5	< 0.01	0.01	< 0.01	< 0.01
b06	13	4	< 0.01	0.07	5.0	< 0.01
s27	6	2	< 0.01	0.01	0.02	< 0.01
s208	16	16	< 0.01	0.32	2.4	< 0.01
s386	13	3	1.0	7.6	8.2	< 0.01
bbara	10	6	0.04	0.01	0.04	< 0.01
beecount	7	3	< 0.01	0.01	0.01	< 0.01
dk14	7	2	45	< 0.01	0.08	< 0.01
donfile	24	3	12316	0.02	1.7	< 0.01

Apply FSM traversal to arithmetic ckts



(a)



(b)

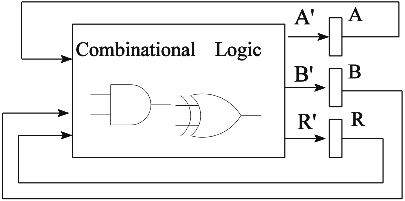
- Model: restricted Moore finite state machine
  - Some sequential arithmetic circuits will give results after running for k clock cycles
  - The initial operands are preloaded in register files
- State transitions on this model:

$$R_k = Tr(R_{k-1}) = Tr(Tr(\cdots Tr(R_{init})\cdots)) = Tr^k(R_{init})$$

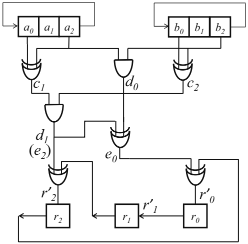
- Word-level unrolling

▸ Recall unrolling

SPEC: $R = A_{init} \cdot B_{init} \pmod{P(\alpha)}$ after k clock cycles



(a)



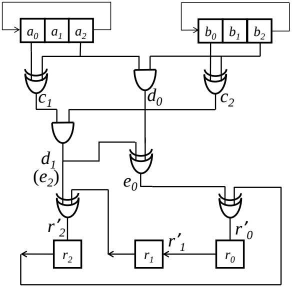
(b)

- Projection on NS R', A', B'

ALGORITHM: Abstraction via implicit unrolling for Sequential GF circuit verification

Input: Circuit polynomial ideal J, vanishing ideal J_0, initial state ideal $R(=0), \mathcal{G}(A_{init}), \mathcal{H}(B_{init})$

$from_0(R, A, B) = \langle R, \mathcal{G}(A_{init}), \mathcal{H}(B_{init})\rangle$;
$i = 0$;
repeat
    $i \leftarrow i + 1$;
    $G \leftarrow \text{GB}(\langle J + J_0 + from_{i-1}(R, A, B)\rangle)$ with ATO;
    $to_i(R', A', B') \leftarrow G \cap \mathbb{F}_{2^k}[R', A', B', R, A, B]$;
    $from_i \leftarrow to_i(\{R, A, B\} \setminus \{R', A', B'\})$;
until $i == k$;
return $from_k(R_{final})$



- The elimination ideal (first iteration):

$$\begin{aligned}J = &d_0 + b_2 \cdot a_2, c_1 + a_0 + a_2, c_2 + b_0 + b_2, d_1 + c_1 \cdot c_2,\\ &e_0 + d_0 + d_1, e_2 + d_1, r'_0 + r_2 + e_0, r'_1 + r_0, r'_2 + r_1 + e_2,\\ &A + a_0\beta + a_1\beta^2 + a_2\beta^4, B + b_0\beta + b_1\beta^2 + b_2\beta^4,\\ &R + r_0\beta + r_1\beta^2 + r_2\beta^4, R' + r'_0\beta + r'_1\beta^2 + r'_2\beta^4;\end{aligned}$$

- $J_0 = \langle x_i^2 - x_i, X^q - X \rangle$
- $from_0 = \{R, A_{init} + a_0\beta + a_1\beta^2 + a_2\beta^4, B_{init} + b_0\beta + b_1\beta^2 + b_2\beta^4\}$

Basic algorithm to verify the function of sequential GF multipliers

ALGORITHM: Abstraction via implicit unrolling for Sequential GF circuit verification

Input: Circuit polynomial ideal J, vanishing ideal J_0, initial state ideal $R(=0), \mathcal{G}(A_{init}), \mathcal{H}(B_{init})$

$from_0(R, A, B) = \langle R, \mathcal{G}(A_{init}), \mathcal{H}(B_{init}) \rangle$;
$i = 0$;
repeat
    $i \leftarrow i + 1$;
    $G \leftarrow$GB($\langle J + J_0 + from_{i-1}(R, A, B) \rangle$) with ATO;
    $to_i(R', A', B') \leftarrow G \cap \mathbb{F}_{2^k}[R', A', B', R, A, B]$;
    $from_i \leftarrow to_i(\{R, A, B\} \setminus \{R', A', B'\})$;
until $i == k$;
return $from_k(R_{final})$

- $J_0 = \langle x_i^2 - x_i, X^q - X \rangle$
- $from_0 = \{R, A_{init} + a_0\beta + a_1\beta^2 + a_2\beta^4, B_{init} + b_0\beta + b_1\beta^2 + b_2\beta^4\}$ $(\beta = \alpha^3)$
- $to_1 : R' + (\alpha^2)A_{init}^4 B_{init}^4 + (\alpha^2+\alpha)A_{init}^4 B_{init}^2 + (\alpha^2+\alpha)A_{init}^4 B_{init} + (\alpha^2+\alpha)A_{init}^2 B_{init}^4 + (\alpha^2+\alpha+1)A_{init}^2 B_{init}^2 + (\alpha^2)A_{init}^2 B_{init} + (\alpha^2+\alpha)A_{init} B_{init}^4 + (\alpha^2)A_{init} B_{init}^2$
- $from_1 = \{R' + (\alpha^2)A_{init}^4 B_{init}^4 + (\alpha^2+\alpha)A_{init}^4 B_{init}^2 + (\alpha^2+\alpha)A_{init}^4 B_{init} + (\alpha^2+\alpha)A_{init}^2 B_{init}^4 + (\alpha^2+\alpha+1)A_{init}^2 B_{init}^2 + (\alpha^2)A_{init}^2 B_{init} + (\alpha^2+\alpha)A_{init} B_{init}^4 + (\alpha^2)A_{init} B_{init}^2, A_{init} + a_2\alpha^3 + a_0\alpha^6 + a_1\alpha^{12}, B_{init} + b_2\alpha^3 + b_0\alpha^6 + b_1\alpha^{12}\}$
- $\cdots$
- After 3 iterations: $to_3 = \{R' + A_{init}B_{init},\ A_{init} + a_0'\alpha^3 + a_1'\alpha^6 + a_2'\alpha^{12},\ B_{init} + b_0'\alpha^3 + b_1'\alpha^6 + b_2'\alpha^{12}\}$

ALGORITHM: Abstraction via implicit unrolling for Sequential GF circuit verification

Input: Circuit polynomial ideal J, vanishing ideal J_0, initial state ideal $R(=0), \mathcal{G}(A_{init}), \mathcal{H}(B_{init})$

$from_0(R, A, B) = \langle R, \mathcal{G}(A_{init}), \mathcal{H}(B_{init})\rangle$;
$i = 0$;
repeat
    $i \leftarrow i + 1$;
    $f_o \xrightarrow{J+J_0+from_{i-1}(R,A,B)}_{+} f_r$ under RATO ;
    $to_i(R', A', B') \leftarrow f_r(\{R', A', B'\} \setminus \{r_0, \ldots, r_{k-1}, a_0, \ldots, a_{k-1}, b_0, \ldots, b_{k-1}\}$;
    $from_i \leftarrow to_i(\{R, A, B\} \setminus \{R', A', B'\})$;
until $i == k$;
return $from_k(R_{final})$

Experiment result: sequential GF multiplier verification

- Run-time for verification of bug-free RH-SMPO circuits for SAT, ABC and BDD based methods. TO = timeout 14 hrs

	Word size of the operands k-bits			
Solver	11	18	23	33
Lingeling	593	*TO*	*TO*	*TO*
ABC	6.24	*TO*	*TO*	*TO*
BDD	0.1	11.7	1002.4	*TO*

- Runtime for verification of bug-free Agnew's and RH-SMPO circuits using our approach

Operand size k		36	60	81	100	131	162
RH-SMPO	#Polys	4716	12960	21870	35600	56592	92826
	Runtime	14.3	213.3	1343	4685	26314	124194
Agnew's SMPO	#Polys	2700	7380	13356	20300	34715	52974
	Runtime	10.2	212.0	2684	4686	56568	119441

Abstraction refinement

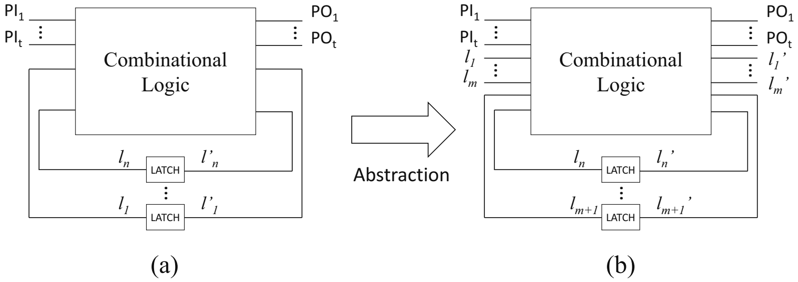


Figure : Abstraction by reducing latches

- Remove "irrelevant" latches, reduce state space
- Provide an over-approximation
- This algorithm requires UNSAT core extraction

Recall *k*-BMC

- Recall Buchberger's algorithm

- Recall Buchberger's algorithm
- What can we learn from the "1" generated by Buchberger's Algorithm?

Theorem

Let $J = \langle f_1, \ldots, f_s \rangle$ be an ideal in the ring $\mathbb{F}[x_1, \ldots, x_d]$ and $V_{\overline{\mathbb{F}}}(J)$ be its variety over $\overline{\mathbb{F}}$. Then $V_{\overline{\mathbb{F}}}(J) = \emptyset \iff J = \mathbb{F}[x_1, \ldots, x_d] \iff 1 \in J$.

$$V_{\overline{\mathbb{F}}}(J) = \emptyset \iff 1 \in J \iff \textit{reduced } GB(J) = \{1\}$$

Motivating Example

- $f_1 \sim f_9 \in \mathbb{F}_2[a, b, c, d]$
- $F = \{f_1, f_2, \ldots, f_9\}$

$$
\begin{aligned}
f_1 &: abc + ab + ac + bc \\
&\quad + a + b + c + 1 \\
f_2 &: b \\
f_3 &: ac \\
f_4 &: ac + a
\end{aligned}
\qquad
\begin{aligned}
f_5 &: bc + c \\
f_6 &: abd + ad + bd + d \\
f_7 &: cd \\
f_8 &: abd + ab + ad + bd + a + b + d + 1 \\
f_9 &: abd + ab + bd + b
\end{aligned}
$$

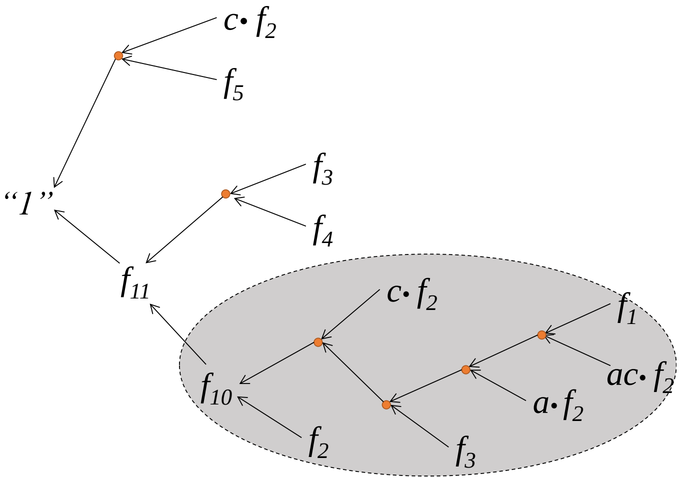
- Find a subset $F_c \subset F$ *s.t.* F_c remains UNSAT

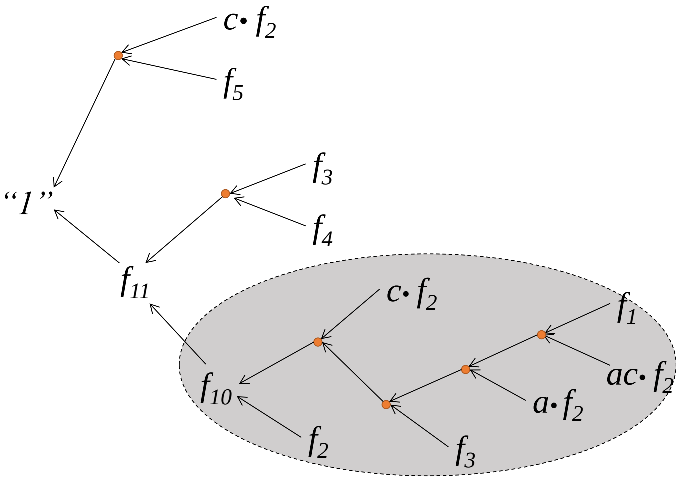
Following Spoly selection strategy $(f_1, f_2) \rightarrow (f_1, f_3) \rightarrow (f_2, f_3) \rightarrow (f_1, f_4) \rightarrow \cdots$, execute Buchberger's algorithm until adding "1" to Gröbner basis

- $Spoly(f_1, f_2) = f_1 - ac \cdot f_2 \xrightarrow{a \cdot f_2} \xrightarrow{f_3} \xrightarrow{c \cdot f_2} \xrightarrow{f_2} f_{10} = a + c + 1$
- $Spoly(f_1, f_3) \xrightarrow{F}_+ 0$
- ...
- $Spoly(f_3, f_4) \xrightarrow{f_{10}} f_{11} = c + 1$
- $Spoly(f_2, f_5) \xrightarrow{f_{11}} 1$

$\{f_1, \ldots, f_9, f_{10}, f_{11}, 1\}$ is the Gröbner basis generated from Buchberger's algorithm

Motivating Example: Refutation Tree





- $f_{10} = f_1 - acf_2 - af_2 - f_3 - cf_2 - f_2 = f_1 + acf_2 + af_2 + f_3 + cf_2 + f_2$
- $1 = \mathbb{F}(f_1, f_2, f_3, f_4, f_5) \implies 1 \in \langle f_1, f_2, f_3, f_4, f_5 \rangle \implies \mathit{UNSAT}$

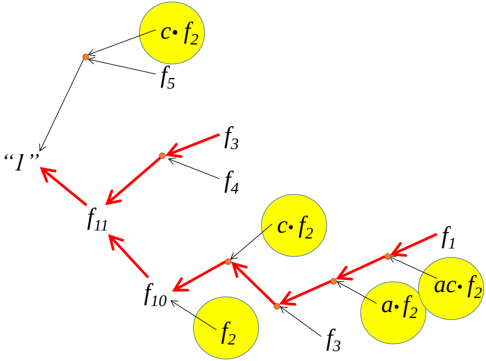
- Expand:

$$
\begin{aligned}
1 &= \mathbb{F}(f_1, f_2, f_3, f_4, f_5) \\
&= cf_2 + f_5 + {\color{red}f_{11}} \\
&= cf_2 + f_5 + f_3 + f_4 + {\color{red}f_{10}} \\
&= (cf_2 + f_5) + \cdots + \mathbf{1} \cdot \mathbf{f_3} + \cdots + \mathbf{1} \cdot \mathbf{f_3} + \cdots + (f_1 + acf_2)
\end{aligned}
$$

- UNSAT core reduced to $\{f_1, f_2, f_4, f_5\}$ which is **minimal**
- GB-core algorithm:
  - Execute Buchberger's algorithm
  - Recording data including Spoly, polynomials for division and remainder
  - Terminate Buchberger's algorithm after recording remainder "1"
  - Build refutation tree and get UNSAT core
  - Analyze recorded data, remove redundant polynomials from the core

Reducing Size: Iterative Refinement

- Spoly pair selection strategy:
  $(f_1, f_2) \to (f_1, f_3) \to (f_2, f_3) \to (f_1, f_4) \to (f_2, f_4) \to \cdots$
- High likelihood in minimal core $\to$ Put ahead in Spoly queue $\to$ Faster approaching "1" in GB-core $\to$ Smaller core

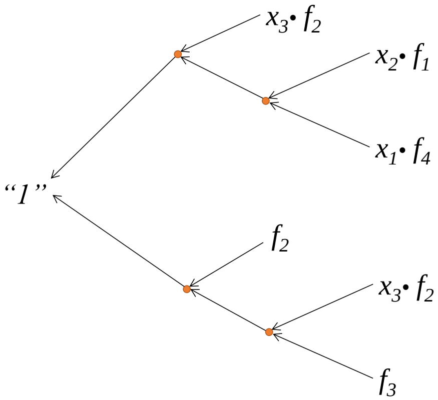


- **Refutation distance**: shortest path to leaf
- Distance$\downarrow \to$ LT Degree$\downarrow \to$ Likelihood in minimal core $\uparrow$
- **Frequency**: number of times f_i appears in refutation tree

Iterative Refinement Example

$f_1 : x_1x_3 + x_3$ $f_2 : x_2 + 1$ $f_3 : x_2x_3 + x_2$

$f_4 : x_2x_3$ $f_5 : x_2x_3 + x_2 + x_3 + 1$ $f_6 : x_1x_2x_3 + x_1x_3$



- $F = \{f_1, f_2, \ldots, f_6\} \in \mathbb{F}_2[x_1, x_2, x_3]$
- UNSAT core: f_1, f_2, f_3, f_4
- Refutation distance:

f_2	f_1	f_3	f_4
2	3	3	3

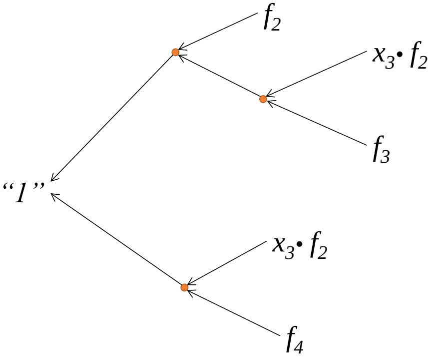
- Frequency:

f_1	f_3	f_4
1	1	1

- Reorder: f_2, f_1, f_3, f_4

$f_1 : x_1x_3 + x_3 \quad f_2 : x_2 + 1 \quad f_3 : x_2x_3 + x_2$

$f_4 : x_2x_3 \quad f_5 : x_2x_3 + x_2 + x_3 + 1 \quad f_6 : x_1x_2x_3 + x_1x_3$



- Initial order: f_2, f_1, f_3, f_4
- Spoly pairs selection: $(f_2, f_1) \rightarrow (f_2, f_3) \rightarrow (f_1, f_3) \rightarrow (f_2, f_4) \rightarrow (f_1, f_4) \rightarrow (f_3, f_4)$
- UNSAT core: f_2, f_3, f_4
- Fixpoint reached

- Finding interdependencies: $f_i \in \langle F \setminus \{f_i\} \rangle$?
- Given $F = \{f_1, \ldots, f_s\}$, find $f_i = \sum_{j \neq i} h_j f_j$
- Info lost in Buchberger's algorithm?
- Inner loop of Buchberger's algorithm:

  $Spoly(f, g) \xrightarrow{G'}_+ r$
  IF $r \neq 0$ THEN $G := G \cup \{r\}$
  IF $r = 0$ THEN Discard
- We collect division info when $r = 0$:
  - Record data for Spoly and polynomial division as in GB-core algorithm
- $Spoly(f_i, f_j) \xrightarrow{F}_+ 0 \implies c_1 f_1 + c_2 f_2 + \cdots + c_s f_s = 0$

Reducing Size Further using Syzygy Heuristic

- Finding interdependencies: $f_i \in \langle F \setminus \{f_i\}\rangle$?
- Given $F = \{f_1, \ldots, f_s\}$, find $f_i = \sum_{j \neq i} h_j f_j$
- Info lost in Buchberger's algorithm?
- Inner loop of Buchberger's algorithm:

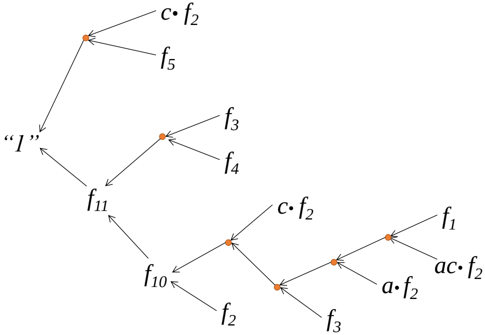
  $Spoly(f, g) \xrightarrow{G'}_+ r$

  IF $r \neq 0$ THEN $G := G \cup \{r\}$

  IF $r = 0$ THEN Discard
- We collect division info when $r = 0$:
  - Record data for Spoly and polynomial division as in GB-core algorithm
- $Spoly(f_i, f_j) \xrightarrow{F}_+ 0 \implies c_1 f_1 + c_2 f_2 + \cdots + c_s f_s = 0$
- $(c_1, c_2, \ldots, c_s)$ is a **syzygy** on $(f_1, f_2, \ldots, f_s)$

- In a single syzygy, $c_i = 1 \implies f_i = \sum_{j \neq i} h_j f_j$
- In general cases, need to analyze all syzygies recorded
- Collect m syzygies as a system of polynomial equations, or a **Syzygy Matrix**

$$\begin{cases} c_1^1 f_1 + c_2^1 f_2 + \cdots + c_s^1 f_s = 0 \\ c_1^2 f_1 + c_2^2 f_2 + \cdots + c_s^2 f_s = 0 \\ \qquad \vdots \\ c_1^m f_1 + c_2^m f_2 + \cdots + c_s^m f_s = 0 \end{cases} \quad \begin{bmatrix} c_1^1 & c_2^1 & \cdots & c_s^1 \\ c_1^2 & c_2^2 & \cdots & c_s^2 \\ \vdots & \vdots & \ddots & \vdots \\ c_1^m & c_2^m & \cdots & c_s^m \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_s \end{bmatrix} = 0$$



- Recorded in Buchberger's Algo:
  $Spoly(f_1, f_2) \xrightarrow{F}_{+} f_{10}$
  $Spoly(f_3, f_4) \xrightarrow{F}_{+} f_{11}$
  $Spoly(f_2, f_5) \xrightarrow{F}_{+} 1$
- Discarded in Buchberger's Algo:
  $Spoly(f_1, f_3) \xrightarrow{F}_{+} 0$
  $Spoly(f_2, f_3) \xrightarrow{F}_{+} 0$
  ...
  $Spoly(f_1, f_5) \xrightarrow{F}_{+} 0$

Syzygy Example

- Syzygy matrix:

$$
\begin{array}{c|cccccccccc}
 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 & f_9 & f_{10} \\
\hline
\text{Spoly}(f_1, f_3) & 1 & a+c+1 & b+1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\text{Spoly}(f_2, f_3) & 0 & ac & b & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{Spoly}(f_1, f_4) & 1 & c+1 & 1 & b & 0 & 0 & 0 & 0 & 0 & 1 \\
\text{Spoly}(f_2, f_4) & 0 & ac+a & 0 & b & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{Spoly}(f_1, f_5) & 1 & a+c+1 & 0 & 0 & a & 0 & 0 & 0 & 0 & 1
\end{array}
$$

Syzygy Example

- Syzygy matrix:

$$
\begin{array}{r}
\\
\text{Spoly}(f_1, f_3) \\
\text{Spoly}(f_2, f_3) \\
\text{Spoly}(f_1, f_4) \\
\text{Spoly}(f_2, f_4) \\
\text{Spoly}(f_1, f_5)
\end{array}
\begin{array}{c}
\begin{array}{cccccccccc}
f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 & f_9 & \boxed{f_{10}}
\end{array} \\
\begin{bmatrix}
1 & a+c+1 & b+1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & ac & b & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & c+1 & 1 & b & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & ac+a & 0 & b & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & a+c+1 & 0 & 0 & a & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\end{array}
$$

- $J = \langle f_1, f_2, \ldots, f_9 \rangle$
- $f_{10} \in J$

Syzygy Example

- Considering
  $f_{10} = f_1 - acf_2 - af_2 - f_3 - cf_2 - f_2 = f_1 + acf_2 + af_2 + f_3 + cf_2 + f_2$

$$
\begin{array}{r c}
 & \begin{array}{cccccccccc} f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 & f_9 & f_{10} \end{array} \\
\begin{array}{r} \mathrm{Spoly}(f_1, f_3) \\ \mathrm{Spoly}(f_2, f_3) \\ \mathrm{Spoly}(f_1, f_4) \\ \mathrm{Spoly}(f_2, f_4) \\ \mathrm{Spoly}(f_1, f_5) \end{array} &
\begin{bmatrix}
1 & a+c+1 & b+1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & ac & b & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & c+1 & 1 & b & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & ac+a & 0 & b & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & a+c+1 & 0 & 0 & a & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \\
\mathrm{Spoly}(f_1, f_2) & [\, 1 \quad \mathrm{ac{+}a{+}c{+}1} \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \,]
\end{array}
$$

Syzygy Example

$$\begin{array}{r|c} & \begin{matrix} f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 & f_9 \end{matrix} \\ \begin{matrix} \mathrm{Spoly}(f_1, f_3) \\ \mathrm{Spoly}(f_2, f_3) \\ \mathrm{Spoly}(f_1, f_4) \\ \mathrm{Spoly}(f_2, f_4) \\ \mathrm{Spoly}(f_1, f_5) \end{matrix} & \begin{bmatrix} 0 & ac & b & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & ac & b & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & ac+a & 0 & b & 0 & 0 & 0 & 0 & 0 \\ 0 & ac+a & 0 & b & 0 & 0 & 0 & 0 & 0 \\ 0 & ac & 1 & 0 & a & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

- Column f_3 contains "1", means:
  $f_3 = acf_2 + af_5$
- Generalization of this strategy refer to the paper

Overall Approach

ALGORITHM: UNSAT core extraction based on Gröbner basis algorithm

Input: A set of UNSAT polynomials F

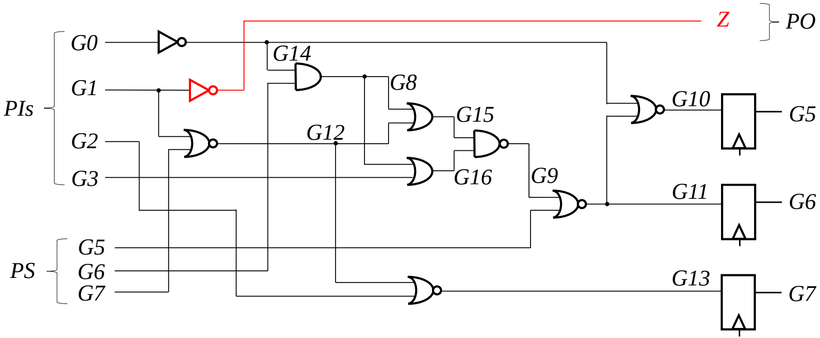
Output: A subset $F' \subset F$ remains UNSAT

$G \leftarrow F$;
repeat
    $F' = G$;
    $G \leftarrow$GB-core(F' with order $>$);
    Update order $>$;
until $G = F'$;
$F' \leftarrow$syzygy_heuristic(G);
return F'

Experiment results: UNSAT core extraction

Table of selected benchmarks
I:single GB-core; II: Iterative GB-core; III: syzygy

Benchmark	# Polys	# MUS	Size of Core			#GB-core iterations	Runtime (sec)			Runtime of PicoMUS (sec)
			I	II	III		I	II	III	
5x5 SMPO	240	137	169	137	137	8	1222	1938	1698	<0.1
aim-100	79	22	22	22	22	1	43	0.7	0.2	<0.1
phole4	104	10	16	16	10	1	4.3	0.2	0.5	<0.1
phole5	169	19	30	25	19	3	12	3.2	2.7	<0.1
subset-2	141	19	37	23	21	2	12	1.6	1.1	<0.1
subset-3	118	16	13	12	11	2	8.6	0.2	0.07	<0.1



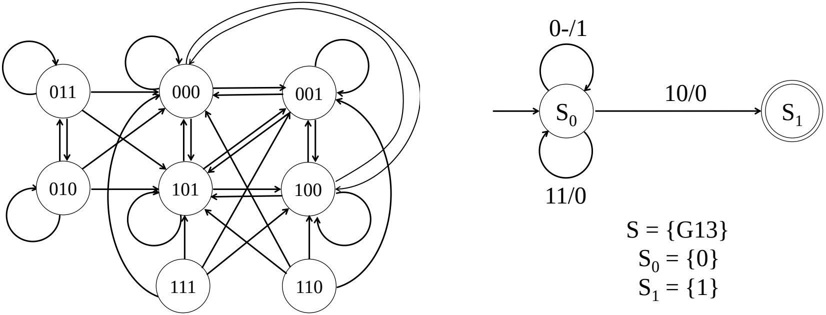
- PS $= \{G7, G6, G5\}$, NS $= \{G13, G11, G10\}$: 8 states
- Property $p = \mathbf{AG}((\neg G13)\mathbf{U}(\neg Z))$
- k-BMC without abstraction refinement: when $k = 3$ prove *PASS*

- Circuit ideal when $k = 0$:

$$
\begin{aligned}
I = \langle & G14 + 1 + G0, G8 + G14 \cdot G6, G15 + G12 + G8 + G12 \cdot G8, \\
& G16 + G3 + G8 + G3 \cdot G8, G9 + 1 + G16 \cdot G15, \\
& G10 + 1 + G14 + G11 + G14 \cdot G11, G11 + 1 + G5 + G9 + G5 \cdot G9, \\
& G12 + 1 + G1 + G7 + G1 \cdot G7, G13 + 1 + G2 + G12 + G2 \cdot G12, \\
& Z + 1 + G1, \\
& (\text{Initial state } 000\ )G5, G6, G7\rangle;
\end{aligned}
$$

- Property: $\neg p = Z \cdot G13 + 1$
- UNSAT core:

$$
\begin{aligned}
Core(I \wedge \neg p) = & G12 + 1 + G1 + G7 + G1 \cdot G7, \\
& G13 + 1 + G2 + G12 + G2 \cdot G12, \\
& Z + 1 + G1, G7;
\end{aligned}
$$



- $\{G5/G10, G6/G11\}$: irrelevant
- By removing irrelevant latches, state-space reduced

Conclusion

- Word-level abstraction of state-space
- Apply to reachability analysis
- Prove effectiveness by experiments on ISCAS'89 and ITC'99 benchmarks

- Word-level abstraction of function in a single time-frame
- Word-level unrolling
- Apply to functional correctness checking of sequential GF multipliers
- Succeed to verify 162-bit, while contemporary fails beyond 23 bit

- UNSAT core extraction for a set of polynomials
- Refine the core using refutation proof & syzygies
- UNSAT core info can be applied to abstraction refinement

Future work

- Multivariate polynomial ideals
  - Extend the application of univariate polynomial ideals
- Accelerate GB reduction
  - F_4 algorithm on term-sparse polynomial ideal (parallel computing)
  - ZDDs can represent chain of OR gates logic in linear space complexity (alternative canonical graphic representation)
- Compute Craig's interpolants using algebraic geometry
  - Projection of varieties $\implies$ interpolants

Publications & tools

- Publications:
  - *Formal Verification of Sequential Galois Field Arithmetic Circuits using Algebraic Geometry*. **Xiaojun Sun**, Priyank Kalla, Tim Pruss, Florian Enescu. DATE 2015, Grenoble
  - *Finding Unsatisfiable Cores of a Set of Polynomials using the Groebner Basis Algorithm*. **Xiaojun Sun**, Irina Ilioaea, Priyank Kalla, Florian Enescu. CP 2016, Toulouse
  - *Word-level Traversal of Finite States Machines using Algebraic Geometry*. **Xiaojun Sun**, Priyank Kalla, Florian Enescu. HLDVT 2016, Santa Cruz
- Tools:
  My website: *ece.utah.edu/∼xiaojuns/code.html*