Finding Unsatisfiable Cores of a Set of Polynomials using the Gröbner Basis Algorithm

Xiaojun Sun[1], Irina Ilioaea[2], Priyank Kalla[1], and Florian Enescu[2]

[1] Electrical and Computer Engineering, University of Utah, Salt Lake City UT, USA
{xiaojuns, kalla}@ece.utah.edu
[2] Mathematics and Statistics, Georgia State University, Atlanta GA, USA
iilioaea1@student.gsu.edu, fenescu@gsu.edu

Abstract. Finding small unsatisfiable subformulas (unsat cores) of infeasible propositional SAT problems is an active area of research. Analogous investigations in the polynomial algebra domain are, however, somewhat lacking. This paper investigates an algorithmic approach to identify a small unsatisfiable core of a set of polynomials, where the corresponding polynomial ideal is found to have an empty variety. We show that such a core can be identified by employing extensions of the Buchberger's algorithm. By further analyzing S-polynomial reductions, we identify certain conditions that are helpful in ascertaining whether or not a polynomial from the given generating set is a part of the unsat core. Our algorithm cannot guarantee a minimal unsat core; the paper describes an approach to refine the identified core. Experiments are performed on a variety of instances using a computer-algebra implementation of our algorithm.

1 Introduction

The Boolean Satisfiability Problem (SAT) is formulated as finding solutions satisfying a set of Boolean equations, or to show that no such solutions exist (unsat). Such problems are often represented in Conjunctive Normal Form (CNF), whereby sets of literal-disjunctions (clauses) must be simultaneously satisfied through some variable assignment. When no solutions exist, it is possible to identify a subset C_c of the original set of clauses C such that C_c is unsat too. This set C_c is called the unsat core of the problem. Unsat cores find application in many areas such as artificial intelligence [1], hardware synthesis [2], formal verification [3], among others. Silva [4] provides a good overview of unsat cores and their applications. An unsat core is a minimally unsatisfiable subformula (MUS) when any of its subsets is satisfiable. Various unsat core extractors are available [5] [6] that can identify smaller/minimal unsat cores from very large unsat problems, and this is an active area of research.

The problem has analogous manifestations in the commutative algebra and algebraic geometry domains. Suppose that we are given a set of polynomials $F = \{f_1, \dots, f_s\}$ with coefficients from any field. Suppose, further, that the system of polynomial equations $f_1 = f_2 = \cdots = f_s = 0$ has no common zero – i.e. the system of polynomial constraints is infeasible (or unsat). Can we identify a subset $F_c \subseteq F$ such that the set of polynomials of F_c also has no common zeros? In this paper, we devise algorithmic techniques to identify such infeasible or unsat cores F_c of a set of polynomials F.

Contributions: Computational algebraic geometry – particularly the theory and technology of Gröbner bases (GB) [7] – provides a mechanism to answer the polynomial unsat core question. The given set of polynomials F generates an ideal, and the celebrated Hilbert's Nullstellensatz [8] provides all the tools to characterize the zero-set (varieties) of polynomial ideals. Most Nullstellensatz-related polynomial decision questions can be answered using Gröbner bases. This motivates our investigation into extraction of infeasible cores of the set of polynomials F using Buchberger's algorithm [9] for GB computation.

We apply Buchberger's algorithm on the given set F to first deduce whether or not F has common zeros, by generating the unit ideal from the polynomials. We perform book-keeping of specific information generated during the execution of Buchberger's algorithm – such as the critical pairs used, the polynomials used in reduction of the basis, quotients of polynomials generated, etc. Analysis of this data allows us to identify the unsat core $F_c \subset F$. We identify certain conditions that help us ascertain whether a polynomial $f_i \in F_c$ can be discarded from the unsat core. Our approach cannot guarantee a minimal core, so we further present a technique to refine the unsat core. Experiments are performed on a variety of hardware equivalence checking and combinatorics benchmarks. In many instances, our approach delivers a minimally infeasible subset F_c.

Previous Work: GB techniques have been employed to tackle SAT problems. Nullstellensatz based proof systems, such as the polynomial calculus [10], used GB as a means to derive proof refutation, and to derive complexity lower bounds [11]. GB techniques have also been used in SAT formula preprocessing [12] and for clause learning [13]. Increasing interest in solving Boolean problems with algebraic reasoning has led to algorithm and tool developments for Boolean Gröbner basis [14] [15]. While investigations of unsat cores have been made for *linear programs* [16] [17], to the best of our knowledge they have not been explored for a *system of polynomial constraints* using Gröbner bases.

The paper is organized as follows. Section 2 covers preliminary concepts. Section 3 describes the basic idea behind our approach. Section 4 describes a method to reduce the size of the infeasible core, along with suitable data-structures and algorithms. Sections 5 and 6 present techniques to further refine the core. Experiments are described in Section 7 and Section 8 concludes the paper.

2 Preliminaries

Let $\mathbb{F}$ be any field, $\overline{\mathbb{F}}$ its algebraic closure, and $R = \mathbb{F}[x_1, \ldots, x_d]$ the polynomial ring in variables $x_1, \ldots, x_d$ with coefficients from $\mathbb{F}$. A monomial in variables $x_1, \ldots, x_d$ is a power product of the form $X = x_1^{e_1} \cdot x_2^{e_2} \cdots x_d^{e_d}$, where $e_i \in \mathbb{Z}_{\geq 0}, i \in \{1, \ldots, d\}$. A *polynomial* $f \in R$ is written as a finite sum of terms $f = c_1X_1 + c_2X_2 + \cdots + c_tX_t$. Here $c_1, \ldots, c_t$ are coefficients and $X_1, \ldots, X_t$ are monomials. To systematically manipulate the polynomials, a monomial order $>$ (or a term order) is imposed on the ring — i.e. a total order on the monomials s.t. multiplication with another monomial preserves the ordering. The monomials of $f = c_1X_1 + c_2X_2 + \cdots + c_tX_t$ are ordered w.r.t. to $>$, such that $X_1 > X_2 > \cdots > X_t$. Subject to $>$, $lt(f) = c_1X_1,\ lm(f) = X_1,\ lc(f) = c_1$, are the *leading term*, *leading monomial* and *leading coefficient* of f, respectively. While our approach works for any permissible term order, in the paper, we consider terms ordered *degree-lexicographically* ($DEGLEX$), where the monomials are ordered according to their descending total degree, with ties broken lexicographically.

Polynomial Reduction: Let f, g be polynomials. If a non-zero term cX of f is divisible by the leading term of g, then we say that f *is reducible to* r modulo g, denoted $f \xrightarrow{g} r$, where $r = f - \frac{cX}{lt(g)} \cdot g$. Similarly, f can be *reduced (divided) w.r.t. a set of polynomials* $F = \{f_1, \ldots, f_s\}$ to obtain a remainder r. This reduction is denoted $f \xrightarrow{F}_+ r$, and the remainder r has the property that no term in r is divisible by the leading term of any polynomial f_i in F. The division algorithm (*e.g.* Alg. 1.5.1 [7]) records the *data* related to both the remainders and the quotients in the division process. We will utilize this data to identify the core.

Ideals, Varieties and Nullstellensatz: Let $F = \{f_1, \ldots, f_s\}$ denote the given set of polynomials. An *ideal* $J \subseteq R$ generated by polynomials $f_1, \ldots, f_s \in R$ is:

$$J = \langle f_1, \ldots, f_s \rangle = \{\sum_{i=1}^{s} h_i \cdot f_i : \ h_i \in \mathbb{F}[x_1, \ldots, x_d]\}.$$

The polynomials $f_1, \ldots, f_s$ form the *basis* or the *generators* of J.

We have to consider the set of solutions to the system of polynomials equations $f_1 = \cdots = f_s = 0$. The set of all solutions to a given system of polynomial equations $f_1 = \cdots = f_s = 0$ is called the *variety*, which depends upon the ideal $J = \langle f_1, \ldots, f_s \rangle$ generated by the polynomials. The variety is denoted by $V(J) = V(f_1, \ldots, f_s)$ and defined as:

$$V(J) = V(f_1, \ldots, f_s) = \{\mathbf{a} \in \mathbb{F}^d : \forall f \in J, f(\mathbf{a}) = 0\},$$

where $\mathbf{a} = (a_1, \ldots, a_d) \in \mathbb{F}^d$ denotes a point in the affine space.

Theorem 1. *The Weak Nullstellensatz [8]: Let* $J = \langle f_1, \ldots, f_s \rangle \subseteq \mathbb{F}[x_1, \ldots, x_d]$ *be an ideal and* $V_{\overline{\mathbb{F}}}(J)$ *be its variety over* $\overline{\mathbb{F}}$. *Then* $V_{\overline{\mathbb{F}}}(J) = \emptyset \iff J = \mathbb{F}[x_1, \ldots, x_d] \iff 1 \in J$.

The Weak Nullstellensatz provides a mechanism to ascertain whether or not a given system of polynomials has a feasible solution. This is deduced by testing whether the unit element is a member of the ideal J. This *ideal membership test* requires the computation of a Gröbner basis.

Definition 1. [**Gröbner Basis**] *[7]: For a monomial ordering* $>$, *a set of non-zero polynomials* $G = \{g_1, g_2, \cdots, g_t\}$ *contained in an ideal* J, *is called a Gröbner basis for* J *iff* $\forall f \in J$, $f \neq 0$ *there exists* $i \in \{1, \cdots, t\}$ *such that* $lm(g_i)$ *divides* $lm(f)$; *i.e.,* $G = GB(J) \Leftrightarrow \ \forall f \in J : f \neq 0, \exists g_i \in G : lm(g_i) \mid lm(f)$. *Equivalently,* $G = \{g_1, g_2, \cdots, g_t\}$ *is a Gröbner basis for* J *iff division by* G *of any polynomial in* J *gives the remainder 0; i.e.* $G = GB(J) \iff \forall f \in J, f \xrightarrow{g_1,\ldots,g_t}_+ 0$.

Buchberger's algorithm [9], shown in Algorithm 1, takes as input the given polynomials $F = \{f_1, \ldots, f_s\}$, and computes the Gröbner basis $G = \{g_1, \ldots, g_t\}$. The algorithm takes pairs of polynomials (f_i, f_j), and computes their *S-polynomial* $(Spoly(f_i, f_j))$:

$$Spoly(f_i, f_j) = \frac{L}{lt(f_i)} \cdot f_i - \frac{L}{lt(f_j)} \cdot f_j$$

where $L = \text{LCM}(lm(f_i), lm(f_j))$. $Spoly(f_i, f_j)$ cancels the leading terms of f_i and f_j. The computation $Spoly(f_i, f_j) \xrightarrow{G'}_+ g_{ij}$ results in a remainder g_{ij}, which if non-zero, provides an element with new leading term in the generating set. The algorithm terminates when for all pairs (f_i, f_j), $Spoly(f_i, f_j) \xrightarrow{G'}_+ 0$.

Algorithm 1: Buchberger's Algorithm

```
**Input**: $F = \{f_1, \ldots, f_s\}$
**Output**: $G = \{g_1, \ldots, g_t\}$
$G := F$;
repeat
    $G' := G$;
    for each pair $\{f_i, f_j\}, i \neq j$ in $G'$ do
        $Spoly(f_i, f_j) \xrightarrow{G'}_+ g_{ij}$ ;
        if $g_{ij} \neq 0$ then
            $G := G \cup \{g_{ij}\}$ ;
        end
    end
until $G = G'$;
```

A Gröbner basis G may contain redundant elements. To remove these redundant elements, G is first made *minimal* and subsequently *reduced.* Subject to the given term order $>$, the reduced GB $G_r = \{g_1, \ldots, g_t\}$ is a unique, canonical representation of ideal J. In the context of Nullstellensatz, $V_{\overline{\mathbb{F}}}(J) = \emptyset \iff G_r = GB(J) = \{1\}$. This implies that for ideals with empty variety, there exists a sequence of $Spoly(f_i, f_j)$ reductions in the reduced GB computation that leads to the unit element. We now show that by analyzing this data, the unsat core F_c of the given generating set $F = \{f_1, \ldots, f_s\}$ can be identified.

3 Motivating the Search for an Unsat Core

With the aforementioned concepts and notations, we can formally state the unsat core problem over polynomial rings as follows:

Problem 1 *Let $F = \{f_1, \ldots f_s\}$ be a set of multivariate polynomials in the ring $R = \mathbb{F}[x_1, \ldots, x_n]$ that generate ideal $J = \langle f_1, \ldots, f_s \rangle \subset R$. Suppose that it is* *known that $V(J) = \emptyset$, or it is determined to be so by applying the Gröbner basis algorithm. Identify a subset of polynomials $F_c \subseteq F, J_c = \langle F_c \rangle$, such that $V(J_c) = \emptyset$ too. Borrowing the terminology from the Boolean SAT domain, we call F_c the infeasible core or the unsat core of F.*

It is not hard to motivate that an unsat core should be identifiable using the Gröbner basis algorithm: Assume that $F_c = F - \{f_j\}$. If $GB(F) = GB(F_c) = \{1\}$, then it implies that f_j is a member of the ideal generated by $(F - \{f_j\})$, i.e. $f_j \in \langle F - \{f_j\} \rangle$. Thus f_j can be composed of the other polynomials of F_c, so f_j is not a part of the unsat core, and it can be safely discarded from F_c. This can be identified by means of the GB algorithm for this ideal membership test.

A näive way (and inefficient way) to identify *a minimal core* using the GB computation is as follows: select a polynomial f_i and see if $V(F_c - \{f_i\}) = \emptyset$ (i.e. if reduced $GB(F_c - \{f_i\}) = \{1\}$). If so, discard f_i from the core; otherwise retain f_i in F_c. Select a different f_i and continue until all polynomials f_i are visited for inclusion in F_c. This approach will produce a minimal core, as we would have tested each polynomial f_i for inclusion in the core. This requires $O(|F|)$ calls to the GB engine, which is really impractical.

3.1 The Refutation Tree of the GB Algorithm: Find F_c from F

We investigate if it is possible to identify a core by analyzing the $Spoly(f_i, f_j) \xrightarrow{F}_+ g_{ij}$ reductions in Buchberger's algorithm. Since F is itself an unsat core, definitely *there exists a sequence of Spoly reductions in Buchberger's algorithm where $Spoly(f_i, f_j) \xrightarrow{F}_+ 1$ is achieved.* Moreover, polynomial reduction algorithms can be suitably modified to record which polynomials from F are used in the division leading to $Spoly(f_i, f_j) \xrightarrow{F}_+ 1$. This suggests that we should be able to identify a core by recording the *"data"* generated by Buchberger's algorithm — namely, the critical pairs(f_i, f_j) used in the Spoly computations, and the polynomials from F used to cancel terms in the reduction $Spoly(f_i, f_j) \xrightarrow{F}_+ 1$. The following example motivates our approach to identify $F_c \subseteq F$ using this "data":

Example 1. Consider the following set of polynomials $F = \{f_1, \dots, f_9\}$:

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

Assume $>_{DEGLEX}$ monomial ordering with $a > b > c > d$. Let $F = \{f_1, \dots, f_9\}$ and $J = \langle F \rangle \subset \mathbb{F}_2[a, b, c, d]$ where $\mathbb{F}_2 = \{0, 1\}$ is the finite field of 2 elements. Then $V(J) = \emptyset$ as $GB(J) = 1$. The set F consists of 4 *minimal* cores: $F_{c1} = \{f_1, f_2, f_3, f_4, f_7, f_8\}, F_{c2} = \{f_2, f_4, f_5, f_6, f_8\}, F_{c3} = \{f_2, f_3, f_4, f_6, f_8\}$, and $F_{c4} = \{f_1, f_2, f_4, f_5\}$.

Buchberger's algorithm terminates to a unique reduced GB, irrespective of the order in which the critical pairs (f_i, f_j) are selected and reduced $Spoly(f_i, f_j) \xrightarrow{F}_+ g_{ij}$. Let us suppose that in the GB computation corresponding to Example 1, the first 3 critical *Spoly* pairs analyzed are $(f_1, f_2), (f_3, f_4)$ and (f_2, f_5). It turns out that the Spoly-reductions corresponding to these 3 pairs lead to the unit ideal. Recording the "data" corresponding to this sequence of reductions is depicted by means of a graph in Fig. 1. We call this graph a refutation tree.

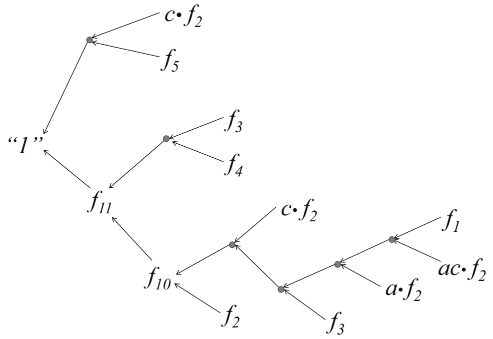

Fig. 1: Generating refutation trees to record unsat cores.

In the figure, the nodes of the graph correspond to the polynomials utilized in Buchberger's algorithm. The leaf nodes always correspond to polynomials from the given generating set. An edge e_{ij} from node i to node j signifies that the polynomial at node j results from polynomial at node i. For example, consider the computation $Spoly(f_1, f_2) \xrightarrow{F}_+ f_{10}$, where $f_{10} = a + c + 1$. Since $Spoly(f_1, f_2) = f_1 - ac \cdot f_2$, the leaves corresponding to f_1 and $ac \cdot f_2$ are created. The reduction $Spoly(f_1, f_2) \xrightarrow{F}_+ f_{10}$ is carried out as the following sequence of 1-step divisions: $Spoly(f_1, f_2) \xrightarrow{a \cdot f_2} \xrightarrow{f_3} \xrightarrow{c \cdot f_2} \xrightarrow{f_2} f_{10}$. This is depicted as the bottom subtree in the figure, terminating at polynomial f_{10}. Moreover, the multiplication $a \cdot f_2$ implies that division by f_2 resulted in the quotient a. The refutation tree of Fig. 1 shows further that $Spoly(f_3, f_4) \xrightarrow{f_{10}} f_{11} = c + 1$ and, finally, $Spoly(f_5, f_2) \xrightarrow{f_{11}} 1$.

To identify an $F_c \subset F$, we start from the refutation node $''1''$, and traverse the graph in reverse, all the way up to the leaves. Then, all the leaves in the transitive fanin of "1" constitute an unsat core. The polynomials (nodes) that do not lie in the transitive fanin of "1" can be safely discarded from F_c. From Fig. 1, $F_c = \{f_1, f_2, f_3, f_4, f_5\}$ is identified as an unsat core of F.

4 Reducing the Size of the Infeasible Core F_c

The core F_c obtained from the aforementioned procedure may contain redundant elements which could be discarded. For example, consider the core $F_c =$

$\{f_1, \dots, f_5\}$ generated in the previous section. While F_c is a smaller infeasible core of F, it is not minimal. In fact, Example 1 shows that $F_{c4} = \{f_1, f_2, f_4, f_5\}$ is the minimal core, where $F_{c4} \subset F_c$. Clearly, the polynomial $f_3 \in F_c$ is a redundant element of the core and can be discarded. We will now describe techniques to further reduce the size of the unsat core by identifying such redundant elements. For this purpose, we will have to perform a more systematic book-keeping of the data generated during the execution of Buchberger's algorithm and the refutation tree.

4.1 Identifying redundant polynomials from the refutation tree

We record the S-polynomial reduction $Spoly(f_i, f_j) \xrightarrow{F}_+ g_{ij}$ that give a non-zero remainder when divided by the system of polynomials F at that moment. The remainder g_{ij} is a polynomial combination of $Spoly(f_i, f_j)$ and the current basis F; thus, it can be represented as:

$$g_{ij} = S(f_i, f_j) + \sum_{k=1}^{m} c_k f_k, \tag{1}$$

where $0 \neq c_k \in \mathbb{F}[x_1, \dots, x_n]$ and $\{f_1, \dots, f_m\}$ is the "current" system of polynomials F. For each non-zero g_{ij}, we will record the following data:

$$((g_{ij})(f_i, h_{ij})(f_j, h_{ji})|(c_{k1}, f_{k1}), (c_{k2}, f_{k2}), \dots, (c_{kl}, f_{kl})) \tag{2}$$

In Eqns. (1) and (2), g_{ij} denotes the remainder of the S-polynomial $Spoly(f_i, f_j)$ modulo the current system of polynomials $f_1, \dots, f_m$, and we denote by

$$h_{ij} := \frac{\text{LCM}(lm(f_i), lm(f_j))}{lt(f_i)}, h_{ji} := -\frac{\text{LCM}(lm(f_i), lm(f_j))}{lt(f_j)}$$

the coefficients of f_i, respectively f_j, in the S-polynomial $Spoly(f_i, f_j)$. Furthermore, in Eqn. (2), $(c_{k1}, \dots c_{kl})$ are the respective quotients of division by polynomials $(f_{k1}, \dots, f_{kl})$, generated during the $Spoly$ reduction.

Example 2. Revisiting Example 1, and Fig. 1, the data corresponding to $Spoly(f_1, f_2) \xrightarrow{F}_+ g_{12} = f_{10}$ reduction is obtained as the following sequence of computations:

$$f_{10} = g_{12} = f_1 - acf_2 - af_2 - f_3 - cf_2 - f_2.$$

As the coefficient field is $\mathbb{F}_2$ in this example, $-1 = +1$, so:

$$f_{10} = g_{12} = f_1 + acf_2 + af_2 + f_3 + cf_2 + f_2$$

is obtained. The data is recorded according to Eqn. (2):

$$((f_{10} = g_{12}), (f_1, 1)(f_2, ac)|(a, f_2), (1, f_3), (c, f_2), (1, f_2))$$

Our approach and the book-keeping terminates when we obtain 1 as the remainder of some S-polynomial modulo the current system of generators. As an output of the Buchberger's algorithm, we can obtain not only the Gröbner basis $G = \{g_1, \ldots, g_t\}$, but also a matrix M such that:

$$\begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_t \end{bmatrix} = M \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_s \end{bmatrix} \tag{3}$$

Each element g_i of G is a polynomial combination of $\{f_1, \ldots, f_s\}$. Moreover, this matrix M is constructed precisely using the data that is recorded in the form of Eqn. (2). We now give a condition when the matrix M may identify some redundant elements.

Theorem 2. *With the notations above, we have that a core for the system of generators $F = \{f_1, \ldots, f_s\}$ of the ideal J is given by the union of those f_i's from F that appear in the data recorded above and correspond to the nonzero entries in the matrix M.*

Proof. In our case, since the variety is empty, and hence the ideal is unit, we have that $G = \{g_1 = 1\}$ and $t = 1$. Therefore $M = [a_1, \ldots, a_s]$ is a vector. Then the output of the algorithm gives: $1 = a_1 f_1 + \cdots + a_s f_s$. Clearly, if $a_i = 0$ for some i then f_i does not appear in this equation and should not be included in the infeasible core of F.

Example 3. Corresponding to Example 1 and the refutation tree shown in Fig. 1, we discover that the polynomial f_3 is used only twice in the division process. In both occasions, the quotient of the division is 1. From Fig. 1, it follows that:

$$1 = (f_2 + f_5) + \cdots + \mathbf{1} \cdot \mathbf{f_3} + \cdots + \mathbf{1} \cdot \mathbf{f_3} + \cdots + (f_1 + f_2) \tag{4}$$

Since $1 + 1 = 0$ over $\mathbb{F}_2$, we have that the entry in M corresponding to f_3 is 0, and so f_3 can be discarded from the core.

4.2 The GB-Core Algorithm Outline

The following steps describe an algorithm (GB-Core) that allows us to compute a refutation tree of the polynomial set and corresponding matrix M.

Inputs: Given a system of polynomials $F = \{f_1, \ldots, f_s\}$, a monomial order $>$ on $\mathbb{F}[x_1, \ldots, x_n]$.

S-polynomial reduction: We start computing the S-polynomials of the system of generators $\{f_1, \ldots, f_s\}$, then divide each of them by the current basis $G = \{f_1, \ldots, f_s, \ldots, f_m\}$, which is the intermediate result of Buchberger's algorithm. In this way, we obtain expressions of the following type:

$$g_{ij} = \underbrace{h_{ij} f_i + h_{ji} f_j}_{Spoly(f_i, f_j)} + \sum_{k=1}^{m} c_k f_k \tag{5}$$

If the remainder g_{ij} is non-zero, we denote it by f_{m+1} and add it to the current set of generators G. We also record the data as in Eqn. (2):

$$((f_{m+1} = g_{ij})(f_i, h_{ij})(f_j, h_{ji})|(c_{k1}, f_{k1}), (c_{k2}, f_{k2}), \ldots, (c_{kl}, f_{kl}))$$

This data forms a part of the refutation tree rooted at node f_{m+1}.

Recording the coefficients: In Eqn. (5) we obtain a vector of polynomial coefficients c_k where $k > s$. These coefficients are associated with new elements (remainders) in the Gröbner basis, that are not a part of the unsat core. Since each polynomial f_k, $(k > s)$ is generated by $\{f_1, \ldots, f_s\}$, we can re-express f_k in terms of $\{f_1, \ldots, f_s\}$. Thus, each $f_k, k > s$ can be written as $f_k = d_1 f_1 + \cdots + d_s f_s$. This process adds a new row $(d_1, \ldots, d_s)$ to the coefficient matrix M.

Termination and refutation tree construction: We perform S-polynomial reductions and record these coefficients generated during the division until the remainder $f_m = 1$ is encountered. The corresponding data is stored in a data-structure D corresponding to Eqn. 2. The matrix M is also constructed. From this recorded data the refutation tree can be easily derived.

We start with the refutation node "$f_m = 1$":

$$((f_m = 1)(f_i, h_{ij})(f_j, h_{ji})|(c_{k1}, f_{k1}), (c_{k2}, f_{k2}), \ldots, (c_{kl}, f_{kl}))$$

and recursively substitute the expressions for the polynomials f_k $(k > s)$ until we obtain the tree with all the leaf nodes corresponding to the original set of polynomials $\{f_1, \ldots, f_s\}$. Algorithm 2 describes this data recording through which the refutation tree T and the matrix M is derived.

Algorithm 2: GB-core algorithm (based on Buchberger's algorithm)

```
Input: $F = \{f_1, \ldots, f_s\} \in \mathbb{F}[x_1, \ldots, x_n], f_i \neq 0$
Output: Refutation tree $T$ and coefficients matrix $M$
Initialize: list $G \leftarrow F$; Dataset $D \leftarrow \emptyset$; $M \leftarrow s \times s$ unit matrix
for each pair $(f_i, f_j) \in G$ do
    $f_{sp}, (f_i, h_{ij})(f_j, h_{ji}) \leftarrow$ Spoly$(f_i, f_j)$ {$f_{sp}$ is the S-polynomial}
    $g_{ij}|(c_{k1}, f_{k1}), \ldots, (c_{kl}, f_{kl}) \leftarrow (f_{sp} \xrightarrow{G}_+ g_{ij})$
    if $g_{ij} \neq 0$ then
        $G \leftarrow G \cup g_{ij}$
        $D = D \cup ((g_{ij})(f_i, h_{ij})(f_j, h_{ji})|(c_{k1}, f_{k1}), (c_{k2}, f_{k2}), \ldots, (c_{kl}, f_{kl}))$
        Update matrix $M$
    end if
    if $g_{ij} = 1$ then
        Construct $T$ from $D$
        Return$(T, M)$
    end if
end for
```

Notice that the core can actually be derived directly from the matrix M. However, we also construct the refutation tree T as it facilitates an iterative refinement of the core, which is described in the next section.

5 Iterative Refinement of the Unsat Core

As with most other unsat core extractors, our algorithm also cannot generate a minimal core in one execution. To obtain a smaller core, we re-execute our algorithm with the core obtained in the current iteration. We describe two heuristics that are applied to our algorithm to increase the likelihood of generating a smaller core in the next iteration.

An effective heuristic should increase the chances that the refutation "1" is composed of fewer polynomials. In our GB-core algorithm, we use a strategy to pick critical pairs such that polynomials with larger indexes get paired *later* in the order:

$$(f_1, f_2) \to (f_1, f_3) \to (f_2, f_3) \to (f_1, f_4) \to (f_2, f_4) \to \cdots$$

Moreover, for the reduction process $Spoly(f_i, f_j) \xrightarrow{F}_+ g_{ij}$, we pick divisor polynomials from F following the increasing order of polynomial indexes. Therefore, by relabeling the polynomial indexes, we can affect their chances of being selected in the unsat core. We use two criteria to to affect the polynomial selection in the unsat core. One corresponds to the *refutation distance*, whereas the other corresponds to the *frequency* with which a polynomial appears in the refutation tree.

Definition 2. Refutation distance of a polynomial f_i *in a refutation tree corresponds to the number of edges on the shortest path from refutation node "1" to any leaf node that represents polynomial* f_i.

On a given refutation tree, polynomials with shorter refutation distances are used as divisors in later stages of polynomial reductions; which implies that they may generally have lower-degree leading terms. This is because we impose a degree-lexicographic term order, and successive divisions (term cancellations) reduce the degree of the remainders. However, what is more desirable is to use these polynomials with lower-degree leading terms earlier in the reduction, as they can cancel more terms. This may prohibit other (higher-degree) polynomials from being present in the unsat core.

Similarly, the motivation for using the *frequency of occurrence* of f_i in the refutation tree is as follows: polynomials that appear frequently in the refutation tree may imply that they have certain properties (leading terms) that give them a higher likelihood of being present in the unsat core.

We apply both heuristics: after the first iteration of the GB-core algorithm, we analyze the refutation tree T and sort the polynomials in the core by the refutation distance criterion, and use the frequency criterion as the tie-breaker. The following example illustrates our heuristic.

Example 4. Consider a set of 6 polynomials over $\mathbb{F}_2$ of an infeasible instance.

$$\begin{aligned} f_1 &: x_1x_3 + x_3; & f_2 &: x_2 + 1 \\ f_3 &: x_2x_3 + x_2; & f_4 &: x_2x_3 \\ f_5 &: x_2x_3 + x_2 + x_3 + 1; & f_6 &: x_1x_2x_3 + x_1x_3 \end{aligned}$$

After the first iteration of the GB-core algorithm, the core is identified as $\{f_1, f_2, f_3, f_4\}$, and we obtain a refutation tree as shown in Fig.2(a).

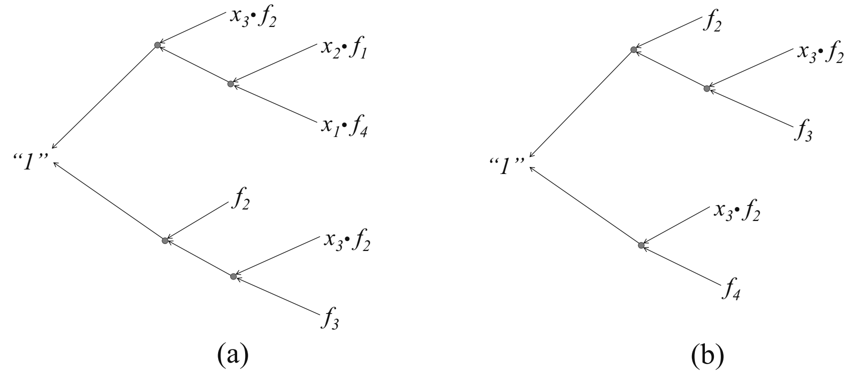

Fig. 2: Refutation trees of core refinement example

The refutation distance corresponding to polynomial f_2 is equal to 2 levels. Note that while three leaf nodes in Fig. 2 (a) correspond to f_2, the shortest distance from “1” to any f_2 node is 2 levels. The refutation distance and frequency measures of other polynomials are identical – equal to 3 and 1, respectively – so their relative ordering is unchanged. We reorder f_2 to be the polynomial with the smallest index. We re-index the polynomial set

$$f'_1 = f_2, f'_2 = f_1, f'_3 = f_3, f'_4 = f_4$$

and apply our GB-core algorithm on the core $\{f'_1, f'_2, f'_3, f'_4\}$. The result is shown in Fig.2(b) with the core identified as $\{f'_1, f'_3, f'_4\} = \{f_2, f_3, f_4\}$. Further iterations do not refine the core – i.e. a fix point is reached.

6 Refining the Unsat core using Syzygies

The unsat core obtained through our GB-core algorithm is by nature a refutation polynomial that equals to 1:

$$1 = \sum_{i=1}^{s} c_i \cdot f_i$$

where $0 \neq c_i \in \mathbb{F}[x_1, \ldots, x_n]$ and the polynomials $F = \{f_1, \ldots, f_s\}$ form a core. Suppose that a polynomial $f_k \in F$ can be represented using a combination of the rest of the polynomials of the core, e.g.:

$$f_k = \sum_{j \neq k} c'_j f_j.$$

Then we can substitute f_k in terms of the other polynomials in the refutation. Thus, f_k can be dropped from the core as it is redundant. One of the limitations of the GB-core algorithm and the re-labeling/refinement strategy is that they cannot easily identify such polynomials f_k in the generating set F that can be composed of the other polynomials in the basis, i.e. $f_k \in \langle F - \{f_k\}\rangle$. We present an approach targeted to identify such combinations to further refine the core.

During the execution of Buchberger's algorithm, many critical pairs (f_i, f_j) do not add any new polynomials in the basis when $Spoly(f_i, f_j) \xrightarrow{F}_+ 0$ gives zero remainder. Naturally, for the purpose of the GB computation, this data is discarded. However, our objective is to gather more information from each GB iteration so as to refine the core. Therefore, we further record the quotient-divisor data from S-polynomial reductions that result in the remainder 0. Every $Spoly(f_i, f_j) \xrightarrow{F}_+ 0$ implies that some polynomial combination of $\{f_1, \dots, f_s\}$ vanishes: i.e. $c_1 f_1 + c_2 f_2 + \cdots + c_s f_s = 0$, for some $c_1, \dots, c_s$. These elements $(c_1, \dots, c_s)$ form a syzygy on $f_1, \dots, f_s$.

Definition 3 (Syzygy [8]). *Let $F = \{f_1, \dots, f_s\}$. A syzygy on $f_1, \dots, f_s$ is an s-tuple of polynomials $(c_1, \dots, c_s) \in (\mathbb{F}[x_1, \dots, x_n])^s$ such that $\sum_{i=1}^{s} c_i \cdot f_i = 0$.*

For each $Spoly(f_i, f_j) \xrightarrow{F}_+ 0$ reduction, we record the information on corresponding syzygies as in Eqn. (6), also represented in matrix form in Eqn. (7):

$$\begin{cases} c_1^1 f_1 + c_2^1 f_2 + \cdots + c_s^1 f_s = 0 \\ c_1^2 f_1 + c_2^2 f_2 + \cdots + c_s^2 f_s = 0 \\ \quad \vdots \\ c_1^m f_1 + c_2^m f_2 + \cdots + c_s^m f_s = 0 \end{cases} \quad (6) \qquad \begin{bmatrix} c_1^1 & c_2^1 & \cdots & c_s^1 \\ c_1^2 & c_2^2 & \cdots & c_s^2 \\ \vdots & \vdots & \ddots & \vdots \\ c_1^m & c_2^m & \cdots & c_s^m \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_s \end{bmatrix} = 0 \quad (7)$$

Here $\{f_1, f_2, \dots, f_s\}$ is the given core. Take one column of the syzygy matrix (e.g. the set of polynomials $c_1^1, c_1^2, \dots, c_1^m$) and compute its reduced Gröbner basis G_r. If $G_r = \{1\}$, then it means that there exists some polynomial vector $r_1, r_2, \dots, r_m$ such that $1 = r_1 c_1^1 + r_2 c_1^2 + \cdots + r_m c_1^m = \sum_{i=1}^{m} r_i c_1^i$. If we multiply each row i in the matrix of Eqn. (7) with r_i, and sum up all the rows, we will obtain the following equation:

$$\begin{bmatrix} 1 & \sum_{i=1}^{m} r_i c_2^i & \cdots & \sum_{i=1}^{m} r_i c_s^i \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_s \end{bmatrix} = 0 \tag{8}$$

This implies that

$$f_1 + \sum_{i=1}^{m} r_i c_2^i f_2 + \sum_{i=1}^{m} r_i c_3^i f_3 + \cdots + \sum_{i=1}^{m} r_i c_s^i f_s = 0,$$

or that f_1 is a polynomial combination of $f_2, \dots, f_s$. Subsequently, we can deduce that f_1 can be discarded from the core. By repeating this procedure, some redundant polynomials can be identified and size of unsat core can be reduced further.

Example 5. Revisiting Example1, execute the GB-core algorithm and record the syzygies on $f_1, \dots, f_s$ corresponding to the S-polynomials that give 0 remainder. The coefficients can be represented as entries in matrix shown below. For example, the first row in the matrix corresponds to the syzygies generated by $Spoly(f_1, f_3) \xrightarrow{F}_+ 0$.

$$\begin{array}{r|cccccccccc|}
 & f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 & f_9 & f_{10} \\
Spoly(f_1,f_3) & 1 & a+c+1 & b+1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
Spoly(f_2,f_3) & 0 & ac & b & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
Spoly(f_1,f_4) & 1 & c+1 & 1 & b & 0 & 0 & 0 & 0 & 0 & 1 \\
Spoly(f_2,f_4) & 0 & ac+a & 0 & b & 0 & 0 & 0 & 0 & 0 & 0 \\
Spoly(f_1,f_5) & 1 & a+c+1 & 0 & 0 & a & 0 & 0 & 0 & 0 & 1
\end{array} \tag{9}$$

Usually, we need to generate extra columns compared to the syzygy matrix of Eqn. (7). In this example, we need to add an extra column for the coefficient of f_{10}. This is because f_{10} is not among the original generating set; however, some S-polynomial pairs require this new remainder f_{10} as a divisor during reduction. In order to remove this extra column, we need to turn the non-zero entries in this column to 0 through standard matrix manipulations.

Recall that we record f_{10} in M as a nonzero remainder when reducing S-polynomial pair $Spoly(f_1, f_2) \xrightarrow{F}_+ f_{10}$. We extract this information from the coefficient matrix M:

$$(1 \quad ac+a+c+1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$$

It represents f_{10} is a combination of f_1 to f_9:

$$f_{10} = f_1 + (ac+a+c+1)f_2 + f_3$$

It can be written in the same syzygy matrix form (with column f_{10} present) as follows:

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

By adding this row vector (Eqn. (10)) to the rows in Eqn. (9) corresponding to the non-zero entries in the column for f_{10}, we obtain the syzygy matrix only for the polynomials in the core:

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

We find out there is a "1" entry in the f_3 column. The last row implies that f_3 is a combination of f_2, f_5 ($f_3 = acf_2 + af_5$), so f_3 can be discarded from the core.

Our Overall Approach for Unsat Core Extraction: i) Given the set $F = \{f_1, \ldots, f_s\}$, we apply the GB-core algorithm, record the data D, M (Section 4) and the syzygies S on $f_1, \ldots, f_s$. ii) From M, we obtain a core $F_c \subseteq F$. iii) Iteratively refine F_c (Section 5) until $|F_c|$ cannot be reduced further. iv) Apply the syzygy-heuristic (Section 6) to identify if some $f_k \in F_c$ is a combination of other polynomials in F_c; all such f_k are discarded from F_c. This gives us the final unsat core F_c.

7 Experiment results

We have implemented our core extraction approach (the GB-Core and the refinement algorithms) using the SINGULAR symbolic algebra computation system [v. 3-1-6] [18]. With our tool implementation, we have performed experiments to extract a minimal unsat core from a given set of polynomials. Our experiments run on a desktop with 3.5GHz Intel Core$^{\text{TM}}$ i7-4770K Quad-core CPU, 16 GB RAM and 64-bit Ubuntu Linux OS. The experiments are shown in Table 1.

Gröbner basis is not an efficient engine for solving contemporary industry-size CNF-SAT benchmarks, as the translation to CNF introduces too many variables and clauses for GB engines to handle. In order to validate our approach, we use a somewhat customized benchmark library: i) "aim-100" is a modified version of the randon 3-SAT benchmark "aim-50/100", modified by adding some redundant clauses; ii) The "subset" series are generated for random subset sum problems; iii) "cocktail" is similarly revised from a combination of factorization and a random 3-SAT benchmark; iv) and "phole4/5" are generated by adding redundant clauses to pigeon hole benchmarks; v) Moreover, SMPO and RH benchmarks correspond to hardware equivalence checking instances of sequential Galois field normal basis modulo multiplier circuits [19] [20], compared against a golden model *spec*. Similarly, the "MasVMon" benchmarks are the equivalence checking circuits corresponding to Mastrovito multipliers compared against Montgomery multipliers [21]. Some of these are available as CNF formulae, whereas others were available directly as polynomials over finite fields. The CNF formulae are translated as polynomial constraints over $\mathbb{F}_2$ (as shown in [12]), and the GB-Core algorithm and the refinement approach is applied.

In Table 1, #Polys denotes the given number of polynomials from which a core is to be extracted. #MUS is the *minimal* core extracted by PicoMUS. #GB-core iterations corresponds to the number of calls to the GB-core engine to arrive at the reduced unsat core. The second last column shows the improvement in the minimal core size by applying the syzygy heuristic on those cases which cannot be iteratively refined further. The data shows that in most of these cases, our tool can produce a minimal core. For the subset-3 benchmark, we obtain a smaller

Table 1: Results of running benchmarks using our tool. Asterisk(*) denotes that the benchmark was not translated from CNF.

Benchmark	# Polys	# MUS	Size of core w/ GB-core + iterative refinement	# GB-core iterations	# Size of core after using syzygy heuristic	Runtime (sec)
5 × 5 SMPO	240	137	137	8	-	3160
4 × 4 SMPO*	84	21	21	1	-	125
3 × 3 SMPO*	45	15	15	1	-	6.6
3 × 3 SMPO	17	2	2	1	-	0.07
4 × 4 MasVMont*	148	83	83	1	-	23
3 × 3 MasVMont*	84	53	53	1	-	4.3
2 × 2 MasVMont	27	23	23	2	-	2.3
5 × 5 RH*	142	34	35	4	35	1001
4 × 4 RH*	104	35	36	3	36	98
3 × 3 RH*	50	20	20	1	-	2.9
aim-100	79	22	22	1	-	43
cocktail	135	4	4	2	-	5
subset-1	100	6	6	1	-	2.4
subset-2	141	19	23	2	21	12
subset-3	118	16	12	2	11	8.7
phole4	104	10	16	1	10	4.4
phole5	169	19	25	3	19	14

core than PicoMUS. The results demonstrate the power of the Gröbner basis technique to identify the causes of unsatisfiability.

8 Conclusions

This paper addressed the problem of identifying an infeasible core of a set of multivariate polynomials, with coefficients from a field, that have no common zeros. The problem is posed in the context of computational algebraic geometry and solved using the Gröbner basis algorithm. We show that by recording the data produced by the Buchberger's algorithm – the $Spoly(f_i, f_j)$ pairs, as well as the polynomials of F used in the division process $Spoly(f_i, f_j) \xrightarrow{F}_+ 1$ – we can identify certain conditions under which a polynomial can be discarded from a core. An algorithm was implemented within the Singular computer algebra tool and some experiments were conducted to validate the approach. While the use of GB engines for SAT solving has a rich history, the problem of unsat core identification using GB-engines has not been addressed by the SAT community. We hope that this paper will kindle some interest in this topic which is worthy of attention from the SAT community – particularly when there seems to be a renewal of interest in the use of Gröbner bases for formal verification [22] [21] [23] [24].

References

1. J. L. de Siqueira and J.-F. Puget, “Explanation-based generalization of failures,” in *Proc. European Conf. Artificial Intelligence*, 1988, pp. 339–344.
2. J.-H. R. Jiang, C.-C. Lee, A. Mishchenko, and C.-Y. Huang, “To SAT or Not to SAT: Scalable Exploration of Functional Dependency,” *IEEE Trans. Comp.*, vol. 59, no. 4, pp. 457–466, April 2010.
3. E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, “Counterexample-Guided Abstraction Refinement for Symbolic Model Checking,” *Journal of ACM*, vol. 50, no. 5, pp. 752–794, Sept 2003.
4. J. Marques-Silva, “Minimal Unsatisfiability: Models, Algorithms and Applications,” in *IEEE Intl. Symp. on Multi-Valued Logic*, 2010, pp. 9–14.
5. A. Nadel, V. Ryvchin, and O. Strichman, “Accelerated deletion-based extraction of minimal unsatisfiable cores,” *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 9, pp. 27–51, 2014.
6. A. Belov, I. Lynce, and J. Marques-Silva, “Towards efficient MUS extraction,” *AI Communications*, vol. 25, no. 2, pp. 97–116, 2012.
7. W. W. Adams and P. Loustaunau, *An Introduction to Gröbner Bases.* American Mathematical Society, 1994.
8. D. Cox, J. Little, and D. O’Shea, *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra.* Springer, 2007.
9. B. Buchberger, “Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal,” Ph.D. dissertation, University of Innsbruck, 1965.
10. M. Clegg, J. Edmonds, and R. Impagliazzo, “Using the Gröbner Basis Algorithm to Find Proofs of Unsatisfiability,” in *ACM Symposium on Theory of Computing*, 1996, pp. 174–183.
11. P. Beame, R. Impagliazzo, J. Kraj’icek, T. Pitassi, and P. Pudl’ak, “Lower bounds on Hilbert’s Nullstellensatz and propositional proofs,” in *Proceedings of the London Mathematical Society*, 1996, pp. 73:1–26.
12. C. Condrat and P. Kalla, “A Gröbner Basis Approach to CNF formulae Preprocessing,” in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2007, pp. 618–631.
13. C. Zengler and W. Kuchlin, “Extending Clause Learning of SAT Solvers with Boolean Gröbner Basis,” in *Intl. Workshop Comp. Algbr. Sci. Comp.*, 2010.
14. M. Brickenstein and A. Dreyer, “Polybori: A Framework for Gröbner Basis Computations with Boolean Polynomials,” *Journal of Symbolic Computation*, vol. 44, no. 9, pp. 1326–1345, September 2009.
15. M. Y. Vardi and Q. Tran, “Groebner Bases Computation in Boolean Rings for Symbolic Model Checking,” in *IASTED*, 2007.
16. J. N. M. van Loon, “Irreducibly inconsistent systems of linear inequalities,” *European Journal of Oper. Res.*, vol. 8, no. 3, pp. 283–288, 1981.
17. J. W. Chinneck and E. Dravnieks, “Locating minimal infeasible constraint sets in linear programs,” *INFORMS Journal on Computing*, vol. 3, no. 2, pp. 157–168, 1991.
18. W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann, “Singular 3-1-3 — A computer algebra system for polynomial computations,” 2011, http://www.singular.uni-kl.de.
19. G. B. Agnew, R. C. Mullin, I. Onyszchuk, and S. A. Vanstone “An implementation for a fast public-key cryptosystem,” *Journal of CRYPTOLOGY*, vol. 3, no. 2, pp. 63–79, 1991.

20. A. Reyhani-Masoleh and M. A. Hasan, "Low complexity word-level sequential normal basis multipliers," *Computers, IEEE Transactions on*, vol. 54, no. 2, pp. 98–110, 2005.
21. J. Lv, P. Kalla, and F. Enescu, "Efficient Gröbner Basis Reductions for Formal Verification of Galois Field Arithmetic Circuits," in *IEEE Trans. on CAD*, vol. 32, no. 9, 2013, pp. 1409–1420.
22. S. Gao, A. Platzer, and E. Clarke, "Quantifier Elimination over Finite Fields with Gröbner Bases," in *Intl. Conf. Algebraic Informatics*, 2011.
23. P. Kalla, "Formal verification of arithmetic datapaths using algebraic geometry and symbolic computation," in *invited tutorial, in Proc. FMCAD*, 2015, p. 2, slides available online http://www.cs.utexas.edu/users/hunt/FMCAD/FMCAD15/slides/fmcad15-tutorial-kalla.pdf.
24. A. Sayed-Ahmed, D. Große, U. Kühne, M. Soeken, and R. Drechsler, "Formal verification of integer multipliers by combining gröbner basis with logic reduction," in *Proc. Design Automation and Test in Europe*, 2016, pp. 1048–1053.