Finding Infeasible 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 constraint problems is an active area of research. The problem has been widely investigated in the propositional SAT domain. Analogous investigations in the polynomial algebra domain are, however, somewhat lacking. This paper investigates an algorithmic approach to identify a small unsatisfiable core of an ideal in polynomial rings with coefficients from any field, where the ideal is found to have an empty variety. We show that such a core can be identified by means the Buchberger's algorithm and its variations. 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; hence the paper discusses opportunities for refinement of the identified core. The approach is demonstrated on a few examples, and experiments are conducted using an implementation of our algorithm within the SINGULAR computer-algebra tool.

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 have various uses in system design and verification, where they can identify the causes of unsat in the problem and restore satisfiability. Various unsat core extractors are available [**?**] [**?**] that can identify smaller cores – and *minimal* unsat cores – from very large unsat problems.

The problem has analogous manifestations in the polynomial algebra domain. 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 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 system of polynomials of F_c also has no common zeros? In other words, we wish to devise algorithmic techniques to identify infeasible or unsat cores F_c of F.

Clearly, computational algebraic geometry – particularly the theory and technology of Gröbner bases – 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 [**?**] provides all the tools to characterize the zero-set (varieties) of polynomial ideals. Most Nullstellensatz-related polynomial decision questions can be answered by the Gröbner basis algorithm [**?**]. This motivates our investigation into extraction of infeasible cores of the set of polynomials F using Gröbner basis techniques.

Previous Work: Moreover, as Gröbner basis techniques find more applications in SAT solving [**?**] [**?**] [**?**], unsat cores of polynomial systems will also find

in hardware and software verification [**?**], as well as

2 Preliminaries

We denote by C the set of clauses representing the CNF-SAT problem. The problem is assumed to be unsat, and C_c denotes the set of clauses that constitute the unsat core.

Let $\mathbb{F}$ be any field, $\overline{\mathbb{F}}$ its algebraic closure, and $R = \mathbb{F}[x_1, \dots, x_d]$ the polynomial ring in variables $x_1, \dots, x_d$ with coefficients from $\mathbb{F}$. A monomial in variables $x_1, \dots, x_d$ is a power product of the form $X = x_1^{e_1} \cdot x_2^{e_2} \cdots x_d^{e_d}$, where $e_i \in \mathbb{Z}_{\geq 0}, i \in \{1, \dots, d\}$. 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 $c_1, \dots, c_t$ are coefficients and $X_1, \dots, X_t$ 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 such a term order, $lt(f) = c_1X_1,\ lm(f) = X_1,\ lc(f) = c_1$, 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*, where the monomials are ordered according to their descending total degree, and ties are 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* $F = \{f_1, \dots, f_s\}$ 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. ??? **??**) 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 is uniquely defined by 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 [?]: 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.

Gröbner bases: An ideal J may have many bases (generators). A *Gröbner basis* (abbreviated as GB) is a generating set for J with special properties that allow to solve many polynomial decision problems, including ideal membership testing. From among the many *equivalent* definitions of a Gröbner basis, we will consider the following two:

Definition 1. [**Gröbner Basis**] *[?]:*
(i) 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., $G = GB(J) \Leftrightarrow \forall f \in J : f \neq 0, \exists g_i \in G : lm(g_i) \mid lm(f)$. Or equivalently,
(ii) $G = \{g_1, g_2, \cdots, g_t\}$ 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 [**?**], shown in Algorithm 1, computes a Gröbner basis over a field. Given polynomials $F = \{f_1, \ldots, f_s\}$, the algorithm computes the Gröbner basis $G = \{g_1, \ldots, g_t\}$. The algorithm takes pairs of polynomials (f, g), and computes their *S-polynomial* ($Spoly(f, g)$):

$$Spoly(f, g) = \frac{L}{lt(f)} \cdot f - \frac{L}{lt(g)} \cdot g$$

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

Algorithm 1: Buchberger's Algorithm

```
Input: $F = \{f_1, \dots, f_s\}$
Output: $G = \{g_1, \dots, g_t\}$
$G := F$;
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
            $G := G \cup \{r\}$ ;
        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.* To obtain a minimal GB, all polynomials g_j are removed from G if there exists a g_i such that $lm(g_i) \mid lm(g_j)$. Then the remaining elements (g_i's) are made monic by dividing each g_i by $lc(g_i)$. This minimal basis is further reduced by ensuring that no term in g_j is divisible by the leading term $lt(g_i)$ for all $i \neq j$. Subject to $>$, the reduced Gröbner basis $G_r = \{g_1, \dots, g_t\}$ is a unique, canonical representation of ideal J.

In the context of Nullstellensatz, $V_{\overline{\mathbb{F}}}(J) = \emptyset \iff G_r = \{1\}$. This implies that for ideals with empty variety, there exists a sequence of *Spoly*(f, g) 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 $F = \{f_1, \dots, f_s\}$ can be identified.

3 Finding the Unsat Core

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

Problem 1 *Let $F = \{f_1, \dots f_s\}$ be a set of multivariate polynomials in the ring $R = \mathbb{F}[x_1, \dots, x_n]$ that generate ideal $J = \langle f_1, \dots, f_s \rangle \subset R$. 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.*

Any set of polynomials F with empty variety is an unsat core in itself. There may be more than one unsat cores in F, some polynomials may be common to multiple cores (i.e. the cores may be non-disjoint), and these cores may have different cardinalities $|F_c|$. The unsat core problems are therefore further classified as:

– Identify a *minimum* core, i.e. a core of minimum cardinality.
– Identify a *minimal* core. An unsat core F_c is minimal with respect to the property that while the variety of F_c is empty, the variety of *any* subset of F_c is *non-empty*, i.e. $V(F_c - \{f_i\}) \neq \emptyset$ for any $f_i \in F_c$.
– Identify any unsat core disregarding minimality; i.e. find any subset F_c of F.

Let us first consider the problem of finding any $F_c \subset F$, disregarding minimality. 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 $f_j \in \langle F - \{f_j\}\rangle$. In other words, 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 (due to Def. 1(ii)).

A näive way (and inefficient way) to identify *a minimal core* using the GB computation is as follows: selectr 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 Investigating algorithmic techniques to find F_c from F

We investigate if it is possible to identify a core by analyzing the $Spoly(f_i, f_j) \xrightarrow{F}_+ r$ reductions in Buchberger's algorithm. Since F is 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 by recording the *"data"* generated by Buchberger's algorithm — i.e. the critical pair polynomials (f_i, f_j) used in the Spoly computations, and the polynomials from F used to cancel terms in the reduction $Spoly(f_i, f_j) \xrightarrow{F}_+ 1$ — we should be able to identify a core.

We will use an example to demonstrate our approach of identifying $F_c \subseteq F$ using this "data":

Example 1. Consider the following set of polynomials:

$$f_1 : abc + ab + ac + bc + a + b + c + 1$$

$$f_2 : b$$

$$f_3 : ac$$

$$f_4 : ac + a$$

$$f_5 : bc + c$$

$$f_6 : abd + ad + bd + d$$

$$f_7 : cd$$

$$f_8 : abd + ab + ad + bd + a + b + d + 1$$

$$f_9 : abd + ab + bd + b$$

Assume $>_{DEGLEX}$ monomial ordering with $a > b > c > d$. Let $F = \{f_1, \ldots, f_9\}$ 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 are selected and reduced as $Spoly(f_i, f_j) \xrightarrow{F}_+ r$. Let us suppose that in the GB computation corresponding to Example 1, the first 3 critical 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 in Fig. 1. We call this graph a refutation tree.

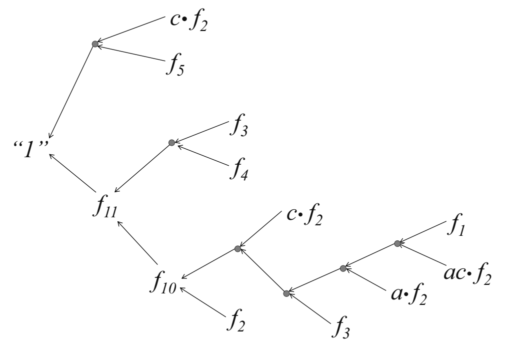

Figure 1: Generating refutation trees to record unsat cores.

In the figure, the nodes of the graph correspond to the polynomials utilized in Buchberger's algorithm. The leaf nodes always correspond to polynomials $f_i \in F$. An edge e_{ij} from node i to node j signifies that the polynomial at node j resulted from polynomial 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 upto 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.

Ordering critical pairs in Buchberger's algorithm: It is clear that the inclusion of polynomials in the obtained core depends on the order in which the critical pairs are reduced. This is a well-known intractable problem, and various heurstics have been proposed to favourably influence the termination of the Gröbner basis algorithm – these include the sugar strategy [?], the preconditioning used by *Faugẽre* in the $F4$ algorithm [?], among others. Any such selection strategy can be also be applied in our setting to identify a smaller core.

4 Reducing the Size of the Infeasible Core F_c

The core F_c obtained from the aforementioned procedure is not guaranteed to be minimal. For example, let us revisit the the core $F_c = \{f_1, \dots, f_5\}$ generated in the previous section. Note that 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. It could be discarded. We will now describe how the size of this core could be reduced further. 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 will keep track of the S-polynomials that give a non-zero remainder when divided by the system of polynomials F at that moment:

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

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

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

Note that in (1) and (2), g_{ij} denotes the remainder of the S-polynomial $S(f_i, f_j)$ modulo the current system of polynomials $f_1, \ldots, f_m$, 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 $S(f_i, f_j)$. Furthermore, in (2), $(c_{k1}, \ldots c_{kl})$ are the polynomial coefficients of $(f_{k1}, \ldots, f_{kl})$ that appear in the division process.

Example 2. Revisiting Example 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 (2):

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

Of course, our approach and the book-keeping stops when we obtain 1 as the remainder of the S-polynomial modulo the current system of generators. As an output of the Buchberger's algorithm, we can obtain not only the Gröbner bases, $G = \{g_1, \ldots, g_t\}$, 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 (2). We now 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 the unit ideal, 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. Therefore, from Fig. 1, it follows that:

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

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 system of generators $G = \{f_1, \ldots, f_s, \ldots, f_m\}$ which is the intermediate result of Buchberger's algorithm. In this way, we obtain the following expression:

$$g_{ij} = h_{ij} f_i + h_{ji} f_j + \sum_{k=1}^{m} c_k f_k \tag{5}$$

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

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

It forms a tree with root f_{m+1}.

Coefficients recording: In (5) we get a coefficient vector including coefficients c_k where $k > s$. These coefficients are associated with new generators in the Gröbner basis, which cannot benefit the UNSAT core extraction. Therefore we need to rewrite the vector to one with only $c_k, k = 1 \ldots s$, which can be achieved by substituting $f_k, k > s$ by $f_1, \ldots, f_s$: initially we have $f_{s+1} = c_1 f_1 + c_2 f_2 + \cdots + (h_{ij} + c_i) f_i + \cdots + (h_{ji} + c_j) f_j + \cdots + c_s f_s$, inductively if we can present f_{s+1} to f_{s+k-1} by $f_1, \ldots, f_s$, we can also rewrite $f_{s+k} = d_1 f_1 + \cdots + d_s f_s$. Then we add a new row $(d_1, \ldots, d_s)$ to coefficient matrix M.

Termination and refutation tree construction: By repeating S-polynomial reduction and coefficients recording until we get remainder $f_m = 1$, the algorithm will generate a set of new generators G and coefficient matrix M. Meanwhile we can construct the refutation tree with the recorded data. We start with

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

It represents a tree start from leaves f_i, f_j then internal nodes $f_{k1}, \dots, f_{kl}$ and "1" as the root. Next we substitute nodes $f_{m-1}, f_{m-2}, \dots, f_{s+1}$ in order with corresponding subtrees. The algorithm returns the complete refutation tree as well as matrix M.

The complete algorithm is concluded as pseudo code below:

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

```
Input: $F = \{f_1, \dots, f_s\} \in \mathbb{F}[x_1, \dots, x_n], f_i \neq 0$
Output: Refutation tree $T$ and coefficients matrix $M$
 1: Initialize: list $G \leftarrow F$; Dataset $D \leftarrow \emptyset$; $M \leftarrow s \times s$ unit matrix
 2: for each pair $(f_i, f_j) \in G$ do
 3:   $f_{sp}, (f_i, h_{ij})(f_j, h_{ji}) \leftarrow$ spoly-pair$(f_i, f_j)$ /* $f_{sp}$ is the S-polynomial */
 4:   $g_{ij}|(c_{k1}, f_{k1}), \dots, (c_{kl}, f_{kl}) \leftarrow (sp \xrightarrow{G}_+ g_{ij})$
 5:   if $g_{ij} \neq 0$ then
 6:     $G \leftarrow G \cup g_{ij}$
 7:     Update $D$ and $M$
 8:   end if
 9:   if $g_{ij} = 1$ then
10:     Construct $T$ from $D$
11:     Return$(T, M)$
12:   end if
13: end for
```

5 Iterative Refinement of the Unsat Core

Similar to all other core extractors, our algorithm cannot generate a minimal core in one execution. To obtain a smaller core, we need to re-execute our tool with the core obtained in the current iteration. We describe two heuristics that are applied to our algorithm to increase the probability to generate a smaller core in the next iteration.

After eliminating all redundant polynomials, we can call our GB engine with the new core. A effective heuristic should enhance the probability that the refutation "1" is composed by fewer polynomials. In our GB computing algorithm, we use a strategy to pick critical pairs such that polynomials with larger indexes get involved as late as possible:

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

Meanwhile, for the reduction process $Spoly(f_i, f_j) \xrightarrow{F}_+ f_{new}$, we pick divisor polynomials from F following the order of polynomial indexes. Therefore, by renaming the polynomial indexes, we can increase the likelihood they are selected into the unsat core. We use 2 criteria to increase the probability that a polynomial is included in the refutation tree. One corresponds to the *refutation distance*, whereas the other corresponds to the frequency that a polynomial appears in refutation tree.

Definition 2. Refutation distance *in a refutation tree corresponds to the number of arcs on the shortest path from refutation "1" to a leaf polynomial.*

Polynomials with shorter refutation distance are used as divisors in later stages of polynomial reductions, which indicates they will generally have lower degree leading terms. This is because we are assuming a degree-lexicographic order, and successive term cancellations reduce the degree of the remainders. So polynomials with lower degree leading terms are more likely to be used for reduction, such that the probability that they appear in the refutation tree is larger.

Similarly, the motivation for *frequency of appearance* of f_i in the refutation tree is as follows: polynomials appearing frequently in the refutation tree also indicates that they have some properties (leading terms) that make them "favourable" in the unsat core selection. For example, their leading terms may contain variables that are require them to be included in the minimal core.

We apply both heuristics: we sort the polynomials in the core by the refutation distance criterion, and use the frequency criterion as the tiebreaker. 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, we get a tree of refutation as shown in Fig.2(a).

The shortest "Distance" corresponds to that of f_2 to "1", which is 2 levels. So, we reorder f_2 to be the polynomial with the shortest index. The "distance" and "frequency" for other polynomials are identical, so we keep their ordering untouched. 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 $\{f_1', f_2', f_3', f_4'\}$. The result is shown in Fig.2(b). We reach a fixpoint, and the core $\{f_1', f_3', f_4'\} = \{f_2, f_3, f_4\}$ is proved to be minimal.

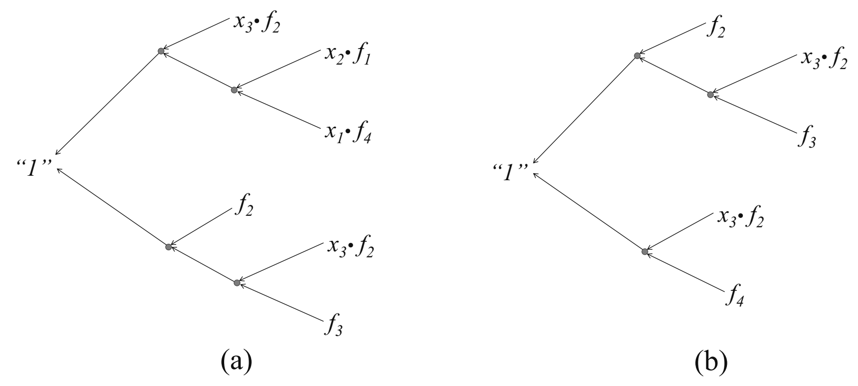

Figure 2: Refutation trees of core refinement example

6 Experiment results

We have implemented our core extraction approach (the GB-Core algorithm) using the SINGULAR symbolic algebra computation system [v. 3-1-6] [?]. 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^{TM} i7-4770K Quad-core CPU, 16 GB RAM and 64-bit Ubuntu Linux OS.

Gröbner basis is not an efficient engine for solving contemporary CNF-SAT benchmarks that are of large size. In order to validate our approach, we use a customized benchmark library: "aim-100" is revised from ramdon 3-SAT benchmark "aim-50/100", "subset" series are generated for random subset sum problems, "cocktail" is revised from a mixed factorization/random 3-SAT benchmark, and "phole4/5" are generated by adding redundandant clauses to pigeon hole benchmarks. Moreover, we created SMPO/RH benchmarks that correspond to equivalence checking instances of sequential Galois field normal basis modulo multiplier implementations (also known as Agnew's multiplier [?] and Reyhani-Hassan multiplier [?] respectively) against the golden model spec. Similarly, the "MasVMon" benchmarks are the miter circuits of Mastrovito multipliers compared against Montgomery multipliers. The CNF formulae are translated as polynomial constraints over $\mathbb{F}_2$ (as shown in [?]), and the GB-Core algorithm is applied.

In Table 1, we list the details of our experimental results. In the table, #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 calling GB-core engine to arrive at an unsatisfiable core. The second last column is showing the improvement of minimal core size applying syzygy heuristic on cases which are not able to be refined further with the iterative heuristic. The data shows that in most of these cases, our tool can produce a minimal core when our iterations converge. From the experimental results

Table 1: Results of running benchmarks using our tool
Asterisk(*) denotes a benchmark which is not converted from CNF benchmark

Benchmark	# Polys	# MUS	Size of core w/ iterative refinement heuristic	# 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		1001
4×4 RH*	104	35	36	3		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

we can also make the observation that the GB-core algorithm, particularly Theorem 2, offers quite a lot of scope for identifying redundant polynomials that can be eliminated from the core — without resorting to a brute-force membership check of every polynomial $f_i \in F - \{f_i\}$.

7 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 hasn't been fully 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.

8 Reduce the size of UNSAT core further

In some cases our iterative core refining algorithm cannot give us the minimal core when it hits the fixpoint. It indicates the limitation of using re-labelling strategy, therefore a new method to identify the redundancy in UNSAT core is needed.

The UNSAT core we obtained through our algorithm is by nature a refutation polynomial that equals to 1:

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

where $c_i \neq 0$ and $f_1, \ldots, f_s$ form a core. If by any means we find a relation between f_k and set $\{f_j\}$ ($f_k, f_j \in \{f_1, \ldots, f_s\}$) and other polynomials in the core such that:

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

Then we can substitute f_k with other polynomials in the refutation polynomial. Thus, f_k can be recognized as redundant polynomial in the core.

Since coefficients $\{c_i\}$ are also polynomials in $\mathbb{F}[x_1, \ldots, x_n]$, we can find enormous number of combinations of $\{f_i\}$. Consider finding an algorithmic solution to this problem without introducing extra search, we propose to utilize the *syzygies* generated during executing the GB-core algorithm.

In Buchberger's algorithm, if the polynomial division gives a nonzero remainder, then the information is recorded. However, if the polynomial division results in a zero remainder nothing will be recorded. In order to gather more information from one iteration, we need to also record all quotients from s-polynomial reductions with 0 remainder. Those quotients (as well as coefficients when pairing s-polynomials) form a syzygy:

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

The matrix form of the syzygy is:

$$\begin{bmatrix} c_1^1 & c_2^1 & \cdots & c_s^1 \\ c_1^2 & c_2^2 & \cdots & c_s^2 \\ \vdots & \vdots & \ddots & \vdots \\ c_1^m & c_2^m & \cdots & c_s^m \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_s \end{bmatrix} = 0 \tag{6}$$

Here $\{f_1, f_2, \ldots, f_s\}$ is the given core. Take one column of the syzygy matrix (e.g. $c_1^1, c_1^2, \ldots, c_1^m$) and compute its minimal Gröbner basis. If the result is 1, then there exists some polynomial vector $r_1, r_2, \ldots, r_m$ 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 the i-th row in above matrix with r_i then add them up, we get following equation

$$\begin{bmatrix} 1 \ \sum_{i=1}^m r_i c_2^i \cdots \ \sum_{i=1}^m r_i c_s^i \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_s \end{bmatrix} = 0 \tag{7}$$

Which implies f_1 is a combination of $f_2, \ldots, f_s$:

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

Then we can remove f_1 from the core. By repeating this procedure, some redundant polynomials can be identified and size of UNSAT core will get closer to minimal.

Example 5. Assume we are given 9 polynomials from Example1, execute GB-core algorithm and record syzygy. We write the coefficients as entries in following matrix:

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

Usually we will get extra columns comparing to syzygy matrix in (6). In this example we add an extra column for coefficient of f_{10} since some s-polynomial pairs require new generator f_{10} as divisor during reduction. In order to remove this extra column, we need to turn nonzero entries in this column to 0 through matrix manipulations. Recall that we record f_{10} as nonzero remainder when reducing s-polynomial pair $Spoly(f_1, f_2)$, we extract corresponding submatrix from the coefficient matrix M

$$(1ac + a + c + 11000000)$$

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 matrix with column f_{10} as follows:

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

By adding this row to corresponding rows in the matrix, we can get syzygy matrix only for polynomials in the core:

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

We find out there is a "1" entry in column f_3, so its minimal GB equals to 1. Therefore f_3 can be recognized as redundant polynomial and removed from the UNSAT core.

References