Quantifier Elimination over Finite Fields Using Gröbner Bases

Sicun Gao, André Platzer, and Edmund Clarke

Carnegie Mellon University, Pittsburgh, PA, USA

Abstract. We give an algebraic quantifier elimination algorithm for the first-order theory over any given finite field using Gröbner basis methods. The algorithm relies on strong Nullstellensatz and properties of elimination ideals over finite fields. We analyze the theoretical complexity of the algorithm and show its application in the formal analysis of a biological controller model.

1 Introduction

We consider the problem of quantifier elimination of first-order logic formulas in the theory T_q of arithmetic in any given finite field F_q, where q is a prime power. Namely, given a quantified formula $\varphi(\boldsymbol{x};\boldsymbol{y})$ in the language, where $\boldsymbol{x}$ is a vector of quantified variables and $\boldsymbol{y}$ a vector of free variables, we describe a procedure that outputs a quantifier-free formula $\psi(\boldsymbol{y})$, such that φ and ψ are equivalent in T_q.

Clearly, T_q admits quantifier elimination. A naive algorithm is to enumerate the exponentially many assignments to the free variables $\boldsymbol{y}$, and for each assignment $\boldsymbol{a} \in F^{|\boldsymbol{y}|}$, evaluate the truth value of the closed formula $\varphi(\boldsymbol{x};\boldsymbol{a})$ (with a decision procedure). Then the quantifier-free formula equivalent to $\varphi(\boldsymbol{x};\boldsymbol{y})$ is $\bigvee_{\boldsymbol{a}\in A}(\boldsymbol{y}=\boldsymbol{a})$, where $A = \{\boldsymbol{a} \in F^{|\boldsymbol{y}|} : \varphi(\boldsymbol{x};\boldsymbol{a}) \text{ is true.}\}$. This naive algorithm *always* requires exponential time and space, and cannot be used in practice. Also, it is important to note that a quantifier elimination procedure is more complex than a decision procedure. The difference is that quantifier elimination yields an equivalent quantifier-free formula (which may contain complex polynomials), while a decision procedure just yields a yes/no answer. For instance, fully quantified formulas over finite fields (such as $\varphi(\boldsymbol{x};\boldsymbol{a})$) can be "bit-blasted" and encoded as Quantified Boolean Formulas (QBF), whose truth value can be decided by QBF solvers (in principle). However, such an encoding only allows us to use QBF decision procedures as an intermediate step in the naive algorithm above, and does not avoid the exponential enumeration of values for the free variables $\boldsymbol{y}$. We believe there has been no investigation into quantifier elimination procedures that can be practically used for this theory.

Such procedures are needed, for instance, in the formal verification of cipher programs involving finite field arithmetic [17, 7] and polynomial dynamical systems over finite fields that arise in systems biology [9, 10, 4]. Take the S2VD virus competition model [9] as an example, which we study in detail in Section 6: The dynamics of the system is given by a set of polynomial equations over the field F_4. We can encode image computation and invariant analysis problems as quantified formulas, which are solvable using quantifier elimination. As is mentioned in [9], there exists no verification method suitable for such systems over general finite fields so far.

In this paper we give an algebraic quantifier elimination algorithm for T_q. The algorithm relies on strong Nullstellensatz and Gröbner basis methods. We analyze its theoretical complexity, and show its applicability in practical problems.

In Section 3, we exploit the strong Nullstellensatz over finite fields and properties of elimination ideals, to show that Gröbner basis computation gives a way of eliminating quantifiers in formulas of the form $\exists \boldsymbol{x}(\bigwedge_i \alpha_i)$, where the α_is are atomic formulas and $\exists \boldsymbol{x}$ is a quantifier block. We then show, in Section 4, that the DNF-expansion of formulas (a common issue in quantifier elimination procedures [16, 15]) can be avoided by using standard ideal operations to "flatten" the formulas. Any quantifier-free formula can be transformed into conjunctions of atomic formulas at the cost of introducing existentially quantified variables. This transformation is linear in the size of the formula, and can be seen as a generalization of the *Tseitin transformation.* Combining the techniques, we obtain a complete quantifier elimination algorithm.

In Section 5, we analyze the complexity of our algorithm, which depends on the complexity of Gröbner basis computation over finite fields. For ideals in $F_q[\boldsymbol{x}]$ that contain $x_i^q - x_i$ for each x_i, Buchberger's Algorithm computes Gröbner bases within exponential time and space [11]. Using this result, the worst-case time/space complexity of our algorithm is bounded by $q^{O(|\varphi|)}$ when φ contains no more than two *alternating blocks* of quantifiers, and $q^{q^{O(|\varphi|)}}$ for more alternations. Recently a polynomial-space algorithm for Gröbner basis computation over finite fields is proposed in [18], but it remains theoretical so far. If the new algorithm can be practically used, the worst-case complexity of quantifier elimination is $q^{O(|\varphi|)}$ for arbitrary alternations. Note that this seemingly high worst-case complexity, as is common for Gröbner basis methods, does not prevent the algorithm from being useful on practical problems. This is crucially different from the naive algorithm, which always requires exponential cost, not just in worst cases. In Section 6, we show how the algorithm is successfully applied in the analysis of a controller design in the S2VD virus competition model [9], which is a polynomial dynamical system over finite fields. The authors developed control strategies to ensure a safety property in the model, and used simulations to conclude that the controller is effective. However, using the quantifier elimination algorithm, we found bugs that show inconsistency between specifications of the system and its formal model. This shows how our algorithm can provide a practical way of extending formal verification techniques to models over finite fields.

Note: Throughout the paper, all the omitted proofs are contained in Appendix A.

2 Preliminaries

2.1 Ideals, Varieties, Nullstellensatz, and Gröbner Bases

Let k be any field and $k[x_1, ..., x_n]$ the polynomial ring over k with indeterminates $x_1, ..., x_n$. An *ideal* generated by $f_1, ..., f_m \in k[x_1, ..., x_n]$ is $\langle f_1, ..., f_m\rangle = \{h : h = \sum_{i=1}^{m} g_i f_i,\ g_i \in k[x_1, ..., x_n]\}$. Let $\boldsymbol{a} \in k^n$ be an arbitrary point, and $f \in k[x_1, ..., x_n]$ be a polynomial. We say that f *vanishes* on $\boldsymbol{a}$ if $f(\boldsymbol{a}) = 0$.

Definition 2.1. *For any subset J of $k[x_1, ..., x_n]$, the* ***affine variety*** *of J over k is*

$$V_n(J) = \{\boldsymbol{a} \in k^n : \forall f \in J, f(\boldsymbol{a}) = 0\}.$$

Definition 2.2. *For any subset V of k^n, the* ***vanishing ideal*** *of V is defined as*

$$I(V) = \{f \in k[x_1, ..., x_n] : \forall \boldsymbol{a} \in V, f(\boldsymbol{a}) = 0\}.$$

Definition 2.3. *Let J be any ideal in $k[x_1, ..., x_n]$, the* ***radical*** *of J is defined as:*

$$\sqrt{J} = \{f \in k[x_1, ..., x_n] : \exists m \in \mathbb{N}, f^m \in J\}.$$

When $J = \sqrt{J}$, we say J is a radical ideal. The celebrated Nullstellensatz established the correspondence between radical ideals and varieties:

Theorem 2.1 (Strong Nullstellensatz [12]). *For an arbitrary field k, let J be an ideal in $k[x_1, ..., x_n]$. We have $I(V^a(J)) = \sqrt{J}$, where k^a is the algebraic closure of k and $V^a(J) = \{\boldsymbol{a} \in (k^a)^n : \forall f \in J, f(\boldsymbol{a}) = 0\}$.*

The methods of Gröbner bases were introduced by Buchberger [5]. For an ideal $\langle f_1, ..., f_m\rangle$, Gröbner basis computation transforms $f_1, ..., f_m$ to a canonical representation $\langle g_1, ..., g_s\rangle = \langle f_1, ..., f_m\rangle$ that has various useful properties (cf. [3]).

Definition 2.4. *Let $T = \{x_1^{\alpha_1} \cdots x_n^{\alpha_n} : \alpha_i \in N\}$ be the set of monomials in $k[x_1, ..., x_n]$. A* ***monomial ordering*** *$\prec$ on T is a well-ordering on T satisfying*
(1) For any $t \in T$, $1 \prec t$
(2) For all $t_1, t_2, s \in T$, $t_1 \prec t_2$ then $t_1 \cdot s \prec t_2 \cdot s$.

We order the monomials appearing in any single polynomial $f \in k[x_1, ..., x_n]$ with respect to $\prec$. We write $LM(f)$ to denote the *leading monomial* in f (the maximal monomial under $\prec$). We write $LT(f)$ to denote the *leading term* of f ($LM(f)$ multiplied by its coefficient). For a set S of polynomials, $LM(S) = \{LM(f) : f \in S\}$.

Definition 2.5. *Let I be an ideal in $k[x_1, ..., x_n]$. Fix any monomial order on T.* ***The ideal of leading monomials*** *of I, $\langle LM(I)\rangle$, is the ideal generated by the leading monomials of all polynomials in I.*

Definition 2.6 (Gröbner Basis). *Let J be an ideal in $k[x_1, ..., x_n]$. A* ***Gröbner basis*** *for J is a set $GB(J) = \{g_1, ..., g_s\} \subseteq J$ satisfying $\langle LM(GB(J))\rangle = \langle LM(J)\rangle$.*

2.2 The First-order Theory over a Finite Field

Let F_q be an arbitrary finite field of size q, where q is a prime power. We fix the structure to be $M_q = \langle F_q, 0, 1, +, \times\rangle$ and the signature $\mathcal{L}_q = \langle 0, 1, +, \times\rangle$ ("=" is a logical predicate). For quantified formulas, we write $\varphi(\boldsymbol{x}; \boldsymbol{y})$ to emphasize that the $\boldsymbol{x}$ is a vector of quantified variables and $\boldsymbol{y}$ is a vector of free variables.

The standard first-order theory for each M_q consists of the usual axioms for fields [14] plus $\exists x_1 \cdots \exists x_q((\bigwedge_{1\leq i<j\leq q} x_i \neq x_j) \wedge \forall y(\bigvee_i y = x_i))$, which fixes the size of the domain. We write this theory as T_q. In $\mathcal{L}_q$, we consider all the atomic formulas as polynomial equations $f = 0$. The *realization* of a formula is the set of assignments to its free variables that makes the formula true over M_q. Formally:

Definition 2.7 (Realization). *Let $\varphi(x_1, ..., x_n)$ be a formula with free variables $\boldsymbol{x} = (x_1, ..., x_n)$. The realization of φ, written as $[\![\varphi]\!] \subseteq F_q^n$, is inductively defined as:*
- $[\![p = 0]\!] =_{df} V(\langle p\rangle) \subseteq F_q^n$ *(in particular,* $[\![\top]\!] = F_q^n$*)*
- $[\![\neg\psi]\!] = F_q^n \setminus [\![\psi]\!]$
- $[\![\psi_1 \wedge \psi_2]\!] = [\![\psi_1]\!] \cap [\![\psi_2]\!]$
- $[\![\exists x_0.\psi(x_0, \boldsymbol{x})]\!] = \{\langle a_1, ..., a_n\rangle \in F_q^n : \exists a_0 \in F_q,$ *such that* $\langle a_0, ..., a_n\rangle \in [\![\psi]\!]\}$

Proposition 2.1 (Fermat's Little Theorem). *Let F_q be a finite field. For any $a \in F_q$, we have $a^q - a = 0$. Conversely, $V(x^q - x) = [\![x^q - x]\!] = F_q$.*

Definition 2.8 (Quantifier Elimination). *T_q admits quantifier elimination if for any quantified formula $\varphi(\boldsymbol{x}; \boldsymbol{y})$, where the $\boldsymbol{x}$ variables are quantified and the $\boldsymbol{y}$ variables free, there exists a quantifier-free formula $\psi(\boldsymbol{y})$ such that $[\![\varphi(\boldsymbol{x}; \boldsymbol{y})]\!] = [\![\psi(\boldsymbol{y})]\!]$.*

2.3 Nullstellensatz in Finite Fields

The Nullstellensatz admits a special form over arbitrary finite fields. This generalizes a result in [8], which was proved for prime fields only and was used in [4, 13]. We give a simple proof in Appendix A, showing it as a corollary of the general strong Nullstellensatz.

Lemma 2.1. *For any ideal $J \subseteq F_q[x_1, ..., x_n]$, $J + \langle x_1^q - x_1, ..., x_n^q - x_n\rangle$ is radical.*

Theorem 2.2 (Strong Nullstellensatz in Finite Fields). *For an arbitrary finite field F_q, let $J \subseteq F_q[x_1, ..., x_n]$ be an ideal, then $I(V(J)) = J + \langle x_1^q - x_1, ..., x_n^q - x_n\rangle$.*

3 Quantifier Elimination Using Gröbner Bases

In this section, we show that the key step in quantifier elimination can be realized by Gröbner basis computation. Namely, for any formula φ of the form

$\exists \boldsymbol{x} \bigwedge_{i=1}^{r} f_i(\boldsymbol{x}, \boldsymbol{y}) = 0$, we can compute a quantifier-free formula $\psi(\boldsymbol{y})$ such that $[\![\varphi(\boldsymbol{x}; \boldsymbol{y})]\!] = [\![\psi(\boldsymbol{y})]\!]$. We use the following notational conventions:

- $|\boldsymbol{x}| = n$ is the number of quantified variables and $|\boldsymbol{y}| = m$ the number of free variables. We write $\boldsymbol{x}^q - \boldsymbol{x} =_{df} \{x_1^q - x_1, ..., x_n^q - x_n\}$ and $\boldsymbol{y}^q - \boldsymbol{y} =_{df} \{y_1^q - y_1, ..., y_m^q - y_m\}$, and call these polynomials the *field polynomials* (following [8]).

- We use $\boldsymbol{a} = (a_1, ..., a_n) \in F_q^n$ to denote the assignment for the $\boldsymbol{x}$ variables, and $\boldsymbol{b} = (b_1, ..., b_m) \in F_q^m$ for the $\boldsymbol{y}$ variables. $(\boldsymbol{a}, \boldsymbol{b}) \in F_q^{n+m}$ is a complete assignment for all the variables in φ.

- When we write $J \subseteq F_q[\boldsymbol{x}, \boldsymbol{y}]$ or a formula $\varphi(\boldsymbol{x}; \boldsymbol{y})$, we assume that all the $\boldsymbol{x}, \boldsymbol{y}$ variables do occur in J or φ. We assume that the $\boldsymbol{x}$ variables always rank higher than the $\boldsymbol{y}$ variables in the lexicographic order.

3.1 Existential Quantification and Elimination Ideals

First, we show that eliminating the $\boldsymbol{x}$ variables is equivalent to *projecting* the variety $V(\langle f_1, ..., f_r\rangle)$ from F_q^{n+m} to F_q^m.

Lemma 3.1. *For $f_1, ..., f_r \in F_q[\boldsymbol{x}, \boldsymbol{y}]$, we have $[\![\bigwedge_{i=1}^{r} f_i = 0]\!] = V(\langle f_1, ..., f_r\rangle)$.*

Definition 3.1 (Projection). *The l-th projection mapping is defined as:*

$$\pi_l : F_q^N \to F_q^{N-l}, \pi_l((c_1, ..., c_N)) = (c_{l+1}, ..., c_N)$$

where $l < N$. For any set $A \subseteq F_q^N$, we write $\pi_l(A) = \{\pi_i(\boldsymbol{c}) : \boldsymbol{c} \in A\} \subseteq F_q^{N-l}$.

Lemma 3.2. $[\![\exists \boldsymbol{x} \varphi(\boldsymbol{x}; \boldsymbol{y})]\!] = \pi_n([\![\varphi(\boldsymbol{x}; \boldsymbol{y})]\!])$.

Next, we show that in finite fields, the projection π_n of the variety $V_{n+m}(\langle f_1, ..., f_r\rangle)$ from F_q^{n+m} to F_q^m, is exactly the variety $V_m(\langle f_1, ..., f_r\rangle \cap F_q[\boldsymbol{y}])$.

Definition 3.2 (Elimination Ideal [6]). *Let $J \subseteq F_q[x_1, ..., x_n]$ be an ideal. The l-th **elimination ideal** J_l, for $1 \leq l \leq N$, is the ideal of $F_q[x_{l+1}, ..., x_N]$ defined by $J_l = J \cap F_q[x_{l+1}, ..., x_N]$.*

The following lemma shows that in conjunctions of atomic formulas, adding field polynomials does not change the realization. For $f_1, ..., f_r \in F_q[\boldsymbol{x}, \boldsymbol{y}]$, we have:

Lemma 3.3. $[\![\bigwedge_{i=1}^{r} f_i = 0]\!] = [\![\bigwedge_{i=1}^{r} f_i = 0 \wedge \bigwedge_{x_i \in \boldsymbol{x}} (x_i^q - x_i = 0) \wedge \bigwedge_{y_i \in \boldsymbol{y}} (y_i^q - y_i = 0)]\!]$.

Now we can prove the key equivalence between projection operations and elimination ideals. This requires the use of the strong Nullstellensatz for finite fields.

Lemma 3.4. *Let $J \subseteq F_q[\boldsymbol{x}, \boldsymbol{y}]$ be an ideal which contains the field polynomials for all the variables in J. We have $\pi_n(V(J)) = V(J_n)$.*

Proof. We show inclusion in both directions.
– $\pi_n(V(J)) \subseteq V(J_n)$: For any $\boldsymbol{b} \in \pi_n(V(J))$, there exists $\boldsymbol{a} \in F_q^n$ such that $(\boldsymbol{a}, \boldsymbol{b}) \in V(J)$. That is, $(\boldsymbol{a}, \boldsymbol{b})$ satisfies all polynomials in J; in particular, $\boldsymbol{b}$ satisfies all polynomials in J that only contain the $\boldsymbol{y}$ variables ($\boldsymbol{a}$ is not assigned to variables). Thus, $\boldsymbol{b} \in V(J \cap F_q[\boldsymbol{y}]) = V(J_n)$.
– $V(J_n) \subseteq \pi_n(V(J))$: Let $\boldsymbol{b}$ be a point in F_q^m such that $\boldsymbol{b} \notin \pi_n(V(J))$.

Consider the polynomial $f_{\boldsymbol{b}} = \prod_{i=1}^m (\prod_{c \in F_q \setminus \{b_i\}} (y_i - c))$.

$f_{\boldsymbol{b}}$ vanishes on all the points in F_q^n, except $\boldsymbol{b} = (b_1, ..., b_m)$, since $(y_i - b_i)$ is excluded in the product for all i. In particular, $f_{\boldsymbol{b}}$ vanishes on all the points in $V(J)$, because for each $(\boldsymbol{a}, \boldsymbol{b'}) \in V(J)$, $\boldsymbol{b'}$ must be different from $\boldsymbol{b}$, and $f_{\boldsymbol{b}}(\boldsymbol{a}, \boldsymbol{b'}) = f_{\boldsymbol{b}}(\boldsymbol{b'}) = 0$ (since there are no $\boldsymbol{x}$ variables). Therefore, $f_{\boldsymbol{b}}$ is contained in the vanishing ideal of $V(J)$, i.e., $f_{\boldsymbol{b}} \in I(V(J))$.

Now, Theorem 2.2 shows $I(V(J)) = J + \langle \boldsymbol{x}^q - \boldsymbol{x}, \boldsymbol{y}^q - \boldsymbol{y} \rangle$. Since J already contains the field polynomials, we know $J + \langle \boldsymbol{x}^q - \boldsymbol{x}, \boldsymbol{y}^q - \boldsymbol{y} \rangle = J$, and consequently $I(V(J)) = J$. Since $f_{\boldsymbol{b}} \in I(V(J))$, we must have $f_{\boldsymbol{b}} \in J$. But on the other hand, $f_{\boldsymbol{b}} \in F_q[\boldsymbol{y}]$. Hence $f_{\boldsymbol{b}} \in J \cap F_q[\boldsymbol{y}] = J_n$. But since $f_{\boldsymbol{b}}(\boldsymbol{b}) \neq 0$, $\boldsymbol{b} \notin V(J_n)$. □

3.2 Quantifier Elimination using Elimination Ideals

Lemma 3.4 shows that to obtain the projection of a variety over F_q, we only need to take the variety of the corresponding elimination ideal. In fact, this can be easily done using the Gröbner basis of the original ideal:

Proposition 3.1 (cf. [6]). *Let $J \subseteq F_q[x_1, ..., x_N]$ be an ideal and let G be the Gröbner basis of J with respect to the lexicographic order $x_1 \succ \cdots \succ x_N$. Then for every $1 \leq l \leq N$, $G \cap F_q[x_{l+1}, ..., x_N]$ is a Gröbner basis of the l-th elimination ideal J_l. That is, $J_l = \langle G \rangle \cap F_q[x_{l+1}, ..., x_N] = \langle G \cap F_q[x_{l+1}, ..., x_N] \rangle$.*

Now, putting all the lemmas together, we arrive at the following theorem:

Theorem 3.1. *Let $\varphi(\boldsymbol{x}; \boldsymbol{y})$ be $\exists \boldsymbol{x}.(\bigwedge_{i=1}^r f_i = 0)$ be a formula in $\mathcal{L}_q$, with $f_i \in F_q[\boldsymbol{x}, \boldsymbol{y}]$. Let G be the Gröbner basis of $\langle f_1, ..., f_r, \boldsymbol{x}^q - \boldsymbol{x}, \boldsymbol{y}^q - \boldsymbol{y} \rangle$. Suppose $G \cap F_q[\boldsymbol{y}] = \{g_1, ..., g_s\}$, then we have $[\![\varphi]\!] = [\![\bigwedge_{i=1}^s (g_i = 0)]\!]$.*

Proof. We write $J = \langle f_1, ..., f_r, \boldsymbol{x}^q - \boldsymbol{x}, \boldsymbol{y}^q - \boldsymbol{y} \rangle$ for convenience. First, by Lemma 3.3, adding the polynomials $\boldsymbol{x}^q - \boldsymbol{x}$ and $\boldsymbol{y}^q - \boldsymbol{y}$ does not change the realization:

$$[\![\varphi]\!] = [\![\exists \boldsymbol{x}.(\bigwedge_{i=1}^r f_i = 0)]\!] = [\![\exists \boldsymbol{x}.(\bigwedge_{i=1}^r f_i = 0 \wedge \bigwedge_{i=1}^n (x_i^q - x_i = 0) \wedge \bigwedge_{i=1}^m (y_i^q - y_i = 0))]\!]$$

Next, by Lemma 3.2, the quantification on $\boldsymbol{x}$ corresponds to projecting a variety:

$$[\![\exists \boldsymbol{x}.(\bigwedge_{i=1}^r f_i = 0 \wedge \bigwedge_{i=1}^n (x_i^q - x_i = 0) \wedge \bigwedge_{i=1}^m (y_i^q - y_i = 0))]\!] = \pi_n(V(J)).$$

Using Lemma 3.4, we know that the projection of a variety is equivalent to the variety of the corresponding elimination ideal, i.e., $\pi_n(V(J)) = V(J \cap F_q[\boldsymbol{y}])$.

Now, using the property of Gröbner bases in Proposition 3.1, we know the elimination ideal $\langle G\rangle \cap F_q[\boldsymbol{y}]$ is generated by $G \cap F_q[\boldsymbol{y}]$:

$$V(J \cap F_q[\boldsymbol{y}]) = V(\langle G\rangle \cap F_q[\boldsymbol{y}]) = V(\langle G \cap F_q[\boldsymbol{y}]\rangle) = V(\langle g_1, ..., g_s\rangle)$$

Finally, by Lemma 3.1, an ideal is equivalent to the conjunction of atomic formulas given by the generators of the ideal: $V(\langle g_1, ..., g_s\rangle) = [\![\bigwedge_{i=1}^{s} g_i = 0]\!]$.

Connecting all the equations above, we have shown $[\![\varphi]\!] = [\![\bigwedge_{i=1}^{s} g_i = 0]\!]$. Note that $g_1, ..., g_s \in F_q[\boldsymbol{y}]$ (they do not contain $\boldsymbol{x}$ variables). $\square$

4 Formula Flattening with Ideal Operations

If negations on atomic formulas can be eliminated (to be shown in Lemma 4.1), Theorem 3.1 already gives a direct quantifier elimination algorithm. That is, we can always use duality to make the innermost quantifier block an existential one, and expand the quantifier-free part to DNF. Then the existential block can be distributed over the disjuncts and Theorem 3.1 is applied. However, this direct algorithm *always* requires exponential blow-up in expanding formulas into DNF.

We show that the DNF-expansion can be avoided: Any quantifier-free formula can be transformed into an equivalent formula of the form $\exists \boldsymbol{z}.(\bigwedge_{i=1}^{r} f_i = 0)$, where $\boldsymbol{z}$ are new variables and f_is are polynomials. The key is that Boolean conjunctions and disjunctions can both be turned into additions of ideals; in the latter case new variables need be introduced. This transformation can be done in linear time and space, and is a generalization of the *Tseitin transformation* to finite fields. The detailed algorithms for flattening formulas are contained in Appendix B.

We use the usual definition of ideal addition and multiplication. Let $J_1 = \langle f_1, ..., f_r\rangle$ and $J_2 = \langle g_1, ..., g_s\rangle$ be ideals, and h be a polynomial. Then $J_1 + J_2 = \langle f_1, ..., f_, g_1, ..., g_s\rangle$ and $J_1 \cdot h = \langle f_1 \cdot h, ..., f_r \cdot h\rangle$.

Lemma 4.1 (Elimination of Negations). *Suppose φ is a quantifier free formula in $\mathcal{L}_q$ in NNF and contains k negative atomic formulas. Then there is a formula $\exists \boldsymbol{z}.\psi$, where ψ contains new variables $\boldsymbol{z}$ but no negative atoms, such that $[\![\varphi]\!] = [\![\exists \boldsymbol{z}.\psi]\!]$.*

Lemma 4.2 (Elimination of Disjunctions). *Suppose ψ_1 and ψ_2 are two formulas in variables $x_1, ..., x_n$, and J_1 and J_2 are ideals in $F_q[x_1, ..., x_n]$ satisfying $[\![\psi_1]\!] = V(J_1)$ and $[\![\psi_2]\!] = V(J_2)$. Then, using x_0 as a new variable, we have $[\![\psi_1 \vee \psi_2]\!] = V(J_1) \cup V(J_2) = \pi_0(V(x_0 J_1 + (1 - x_0)J_2))$.*

Theorem 4.1. *For any quantifier-free formula $\varphi(\boldsymbol{x})$ given in NNF, there exists a formula ψ of the form $\exists \boldsymbol{u}, \boldsymbol{v}(\bigwedge_i (f_i(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}) = 0))$ such that $[\![\varphi]\!] = [\![\psi]\!]$. Furthermore, ψ can be generated in time $O(|\varphi|)$, and also $|\boldsymbol{u}| + |\boldsymbol{v}| = O(|\varphi|)$.*

Proof. Since $\varphi(\boldsymbol{x})$ is in NNF, all the negations occur in front of atomic formulas. We first use Lemma 4.1 to eliminate the negations. Suppose there are k negative atomic formulas in φ, we obtain $[\![\varphi]\!] = [\![\exists u_1, ..., u_k.\varphi']\!]$. Now φ' does not contain negations.

We then prove that there exists an ideal $J_{\varphi'}$ for φ' satisfying $\pi_{|\boldsymbol{v}|}(V(J_{\varphi'})) = [\![\varphi']\!]$, where $\boldsymbol{v}$ are the introduced variables (which rank higher than the existing variables in the variable ordering, so that the projection $\pi_{|\boldsymbol{v}|}$ truncates assignments on the $\boldsymbol{v}$ variables).

- If φ' is an atomic formula $f = 0$, then $J_{\varphi'} = \langle f \rangle$;

- If φ' is of the form $\theta_1 \wedge \theta_2$, then $J_{\varphi'} = J_{\theta_1} + J_{\theta_2}$;

- If φ' is of the form $\theta_1 \vee \theta_2$, then $J_{\varphi'} = v_i \cdot J_{\theta_1} + (1 - v_i) \cdot J_{\theta_2}$, where v_i is new.

Note that the new variables are only introduced in the disjunction case, and therefore the number of $\boldsymbol{v}$ variables equals the number of disjunctions. Following Lemma 3.1 and 4.2, the transformation preserves the realization of the formula in each case. Hence, we have $\pi_{\boldsymbol{v}}(V(J_{\varphi'})) = [\![\varphi']\!]$. Writing $J_{\varphi'} = \langle f_1, ..., f_r \rangle$, we know $[\![\varphi]\!] = [\![\exists \boldsymbol{u}.\varphi']\!] = [\![\exists \boldsymbol{u} \exists \boldsymbol{v}. \bigwedge_{i=1}^{r} f_i]\!]$. Notice that the number of rewriting steps is bounded by the number of logical symbols appearing in φ. Hence the transformation is done in time linear in the size of the formula. The number of new variables is equal to the number of negations and disjunctions. $\square$

5 Algorithm Description and Complexity Analysis

We now describe the full algorithm using the following notations:

- The input formula is given by $\varphi = Q_1 \boldsymbol{x}_1 \cdots Q_m \boldsymbol{x}_m \psi$. Each $Q_i \boldsymbol{x}_i$ represents a *quantifier block*, where Q_i is either $\exists$ or $\forall$. Q_i and Q_{i+1} are different quantifiers. We write $\boldsymbol{x} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_m)$. ψ is a quantifier-free formula in $\boldsymbol{x}$ and $\boldsymbol{y}$ given in NNF, where $\boldsymbol{y}$ are free variables.

- We assume the innermost quantifier is existential, $Q_m = \exists$. (Otherwise we apply quantifier elimination on the negation of the formula.)

5.1 Algorithm Description

Section 4 shows that to eliminate existential quantifiers over conjunctions of positive atomic formulas, we only need to eliminate the polynomials containing the quantified variables in its corresponding ideal. In Section 5, we further showed how formulas can be put into conjunctions of positive atoms with new quantified variables. It follows that we can always eliminate the innermost existential quantifiers, and iterate the process by flipping the universal quantifiers into existential ones. We would like to first emphasize some special features of the algorithm:

- In each elimination step, a full *quantifier block* is eliminated. This is desirable in practical problems, which usually contain many variables but few alternating quantifier blocks. For instance, many verification problems are expressible using two blocks of quantifiers ($\forall\exists$-formulas).

- The quantifier elimination step essentially transforms an ideal to another ideal. This corresponds to transforming conjunctions of atomic formulas to conjunctions of new atomic formulas. Therefore, the quantifier elimination steps do not introduce new nesting of Boolean operators.

- The algorithm always directly outputs CNF formulas.

Algorithm 1 Quantifier Elimination for $\varphi = Q_1\boldsymbol{x}_1 \cdots Q_m\boldsymbol{x}_m.\psi$

```
1: Input: $\varphi = Q_1\boldsymbol{x}_1 \cdots Q_m\boldsymbol{x}_m.\psi(\boldsymbol{x}_1, ..., \boldsymbol{x}_m, \boldsymbol{y})$ where $m$ is the number of quantifier alternations, $Q_m x_m$ is an existential block ($Q_m = \exists$), and $\psi$ is in negation normal form.
2: Output: A quantifier-free equivalent formula of $\varphi$
3: Procedure QE($\varphi$)
4: while $m \geq 1$ do
5:     $\exists \boldsymbol{u}.\psi' \leftarrow$ Eliminate_Negations($\psi$)
6:     $\exists \boldsymbol{v}.(f_1 = 0 \wedge \cdots \wedge f_r = 0) \leftarrow$ Formula_Flattening($\psi'$)
7:     $\varphi \leftarrow Q_1\boldsymbol{x}_1 \cdots Q_m\boldsymbol{x}_m \exists \boldsymbol{u} \exists \boldsymbol{v}.(f_1 = 0 \wedge \cdots \wedge f_r = 0)$
8:     $\{g_1, ..., g_s\} = \text{Gröbner\_Basis}(\langle f_1, ..., f_r, \boldsymbol{x}^q - \boldsymbol{x}, \boldsymbol{u}^q - \boldsymbol{u}, \boldsymbol{v}^q - \boldsymbol{v}\rangle)$
9:     if $m = 1$ then
10:        $\varphi \leftarrow g_1 = 0 \wedge \cdots \wedge g_s = 0$
11:        break
12:    end if
13:    $\varphi \leftarrow Q_1\boldsymbol{x}_1 \cdots Q_{m-2}\boldsymbol{x}_{m-2} Q_{m-1}\boldsymbol{x}_{m-1}.(\bigwedge_{i=1}^{s} g_i = 0)$ where $Q_{m-1} = \forall$
14:    $\varphi \leftarrow Q_1\boldsymbol{x}_1 \cdots Q_{m-2}\boldsymbol{x}_{m-2}.(\bigwedge_{i=1}^{s} \neg\exists \boldsymbol{x}_{m-1}(g_i \neq 0))$
15:    for $i = 1$ to $s$ do
16:        $\bigwedge_{j=1}^{t_i} h_{ij} = 0 \leftarrow$ QE($\exists \boldsymbol{x}_{m-1}(g_i \neq 0)$)
17:    end for
18:    $\varphi \leftarrow Q_1\boldsymbol{x}_1 \cdots Q_{m-2}\boldsymbol{x}_{m-2} \bigwedge_{i=1}^{s}(\bigvee_{j=1}^{t_i} h_{ij} \neq 0)$
19:    $m \leftarrow m - 2$
20: end while
21: return $\varphi$
```

A formal description of the full algorithm is given in Algorithm 1. The main steps in the algorithm are explained below. Each loop of the algorithm contains three main steps. In Step 1, φ is flattened; in Step 2, the innermost existential quantifier block is eliminated; in Step 3, the next (universal) quantifier block is eliminated and the process loops back to Step 1. The algorithm terminates either after Step 2 or Step 3, when there are no remaining quantifiers to be eliminated.

Step 1: (Line 5-7)

First, since ψ is in NNF, we use Theorem 4.1 to eliminate the negations and disjunctions in ψ to get $[\![\varphi]\!] = [\![Q_1\boldsymbol{x}_1 \cdots Q_m\boldsymbol{x}_m \exists \boldsymbol{u} \exists \boldsymbol{v}.(\bigwedge_{i=1}^{r} f_i = 0)]\!]$, where $\boldsymbol{u}$ are the variables introduced for eliminating negations (Lemma 4.1), and $\boldsymbol{v}$ are the variables introduced for eliminating disjunctions (Lemma 4.2).

Step 2: (Line 8-12)

Since $Q_m = \exists$, using Theorem 4.1, we can eliminate the variables $\boldsymbol{x}_m, \boldsymbol{u}, \boldsymbol{v}$ simultaneously by computing $\{g_1, ..., g_{r_1}\} = GB(\langle f_1, ..., f_r, \boldsymbol{x}_m^q - \boldsymbol{x}_m, \boldsymbol{u}^q - \boldsymbol{u}, \boldsymbol{v}^q - \boldsymbol{v}, \boldsymbol{y}^q - \boldsymbol{y}\rangle) \cap F_q[\boldsymbol{x}_1, ..., \boldsymbol{x}_{m-1}, \boldsymbol{y}]$. Now we have $[\![\varphi]\!] = [\![Q_1\boldsymbol{x}_1 \cdots Q_{m-1}\boldsymbol{x}_{m-1}.(\bigwedge_{i=1}^{s}(g_i = 0))]\!]$.

If there are no more quantifiers, the output is $\bigwedge_{i=1}^{s}(g_i = 0)$, which is in CNF.

Step 3: (Line 13-18)

Since $Q_{m-1} = \forall$, we distribute the block $Q_{m-1}\boldsymbol{x}_{m-1}$ over the conjuncts:

$$[\![Q_1\boldsymbol{x}_1 \cdots Q_{m-2}\boldsymbol{x}_{m-2}(\bigwedge_{i=1}^{s} \forall \boldsymbol{x}_{m-1}(g_i = 0))]\!] = [\![Q_1\boldsymbol{x}_1 \cdots Q_{m-2}\boldsymbol{x}_{m-2}(\bigwedge_{i=1}^{s} (\neg\exists \boldsymbol{x}_{m-1}\neg(g_i = 0)))]\!]$$

Now we do elimination recursively on $\exists \boldsymbol{x}_{m-1}(\neg g_i = 0)$ for each $i \in \{1, ..., s\}$, which can be done using only Step 1 and Step 2. We obtain:

$$[\![\exists \boldsymbol{x}_{m-1}(\neg g_i = 0)]\!] = [\![\exists \boldsymbol{x}_{m-1}\exists u'.(g_i \cdot u' - 1 = 0)]\!] = [\![\bigwedge_{j=1}^{t_i} h_{ij} = 0]\!] \qquad (\star)$$

and the formula becomes (note that the extra negation is distributed)

$$[\![\varphi]\!] = [\![Q_1\boldsymbol{x}_1 \cdots Q_{m-2}\boldsymbol{x}_{m-2}.(\bigwedge_{i=1}^{s}(\bigvee_{j=1}^{t_i} h_{ij} \neq 0))]\!]. \qquad (\star\star)$$

If there are no more quantifiers left, the output formula is $\bigwedge_{i=1}^{s}(\bigvee_{j=1}^{t_i} h_{ij} \neq 0)$, which is in CNF. Otherwise, $Q_{m-2} = \exists$, and we return to Step 1.

Theorem 5.1 (Correctness). *Let $\varphi(\boldsymbol{x}; \boldsymbol{y})$ be a formula $Q_1\boldsymbol{x}_i \cdots Q_m\boldsymbol{x}_m.\psi$ where $Q_m = \exists$ and ψ is in NNF. Algorithm 1 computes a quantifier-free formula $\varphi'(\boldsymbol{y})$, such that $[\![\varphi(\boldsymbol{x}; \boldsymbol{y})]\!] = [\![\varphi'(\boldsymbol{y})]\!]$ and φ' is in CNF.*

We provide a complete walk-through example of the algorithm in Appendix C.

5.2 Complexity Analysis

The complexity of Gröbner basis computation on ideals in $F_q[\boldsymbol{x}]$ that contain $x_i^q - x_i$ for each variable x_i is known to be bounded by single exponential in the number of variables in time and space. This follows from the complexity result for Gröbner basis computation of zero-dimensional radical ideals [11]. We have also provided a detailed direct proof of this bound in Appendix D.

Proposition 5.1. *Let $J = \langle f_1, ..., f_r, \boldsymbol{x}^q - \boldsymbol{x}\rangle \subseteq F_q[x_1, ..., x_n]$ be an ideal. The time and space complexity of Buchberger's Algorithm is bounded by $q^{O(n)}$, assuming that the length of input $(f_1, ..., f_r)$ is dominated by $q^{O(n)}$.*

Theorem 5.2 (Complexity). *Let φ be the input formula with m quantifier blocks. When $m \leq 2$, the time/space complexity of Algorithm 1 is bounded by $q^{O(|\varphi|)}$. Otherwise, it is bounded by $q^{q^{O(|\varphi|)}}$.*

Proof. The complexity is dominated by Gröbner basis computation, whose complexity is determined by the number of variables occurring in the ideal. When $m \leq 2$, the main loop is executed once, and the number of newly introduced variables is bounded by the original length of the input formula. Therefore, Gröbner basis computations can be done in single exponential time/space. When $m > 2$, the number of newly introduced variables is bounded by the length of the formula obtained from the previous run of the main loop, which can itself be exponential in the number of the remaining variables. In that case, Gröbner basis computation can take double exponential time/space.

Case $m \leq 2$**:**

In Step 1, the number of the introduced $\boldsymbol{u}$ and $\boldsymbol{v}$ variables equals to the number of negations and disjunctions that appear in the φ. Hence the total number of variables is bounded by the length of φ. The flattening takes linear time and space, $O(|\varphi|)$, as proved in Theorem 4.1.

In Step 2, by Proposition 5.1, Gröbner basis computation takes time/space $q^{O(|\varphi|)}$.

In Step 3, the variables $\boldsymbol{x}_m, \boldsymbol{u}, \boldsymbol{v}$ have all been eliminated. The length of each $g_i u' - 1$ (see Formula ($\star$) in Step 3) is bounded by the number of monomials consisting of the remaining variables, which is $O(q^{(|\boldsymbol{y}|+\sum_{i=1}^{m-1}|\boldsymbol{x}_i|)})$ (because the degree on each variable is lower than q). Following Proposition 5.1, Gröbner basis computation on each $g_i u' - 1$ takes time and space $q^{O(|\boldsymbol{y}|+\sum_{i=1}^{m-1}|\boldsymbol{x}_i|)}$, which is dominated by $q^{O(|\varphi|)}$. Also, since the number s of conjuncts is the number of polynomials in the Gröbner basis computed in the previous step, we know s is bounded by $q^{O(|\varphi|)}$. In sum, Step 3 takes $q^{O(|\varphi|)}$ time/space in worst case.

Thus, the algorithm has worst-case time and space complexity $q^{O(|\varphi|)}$ when $m \leq 2$.

Case $m > 2$**:**

When $m > 2$, the main loop is iterated for more than one round. The key change in the second round is that, the initial number of conjunctions and disjunctions in each conjunct could both be exponential in the number of the remaining variables $(\boldsymbol{x}_1, ..., \boldsymbol{x}_{m-2})$. That means, writing the max of t_i as t (see Formula ($\star\star$) in Step 3), both s and t can be of order $q^{O(|\varphi|)}$.

In Step 1 of the second round, the number of the $\boldsymbol{u}$ variables introduced for eliminating the negations is $s \cdot t$. The number of the $\boldsymbol{v}$ variables introduced for eliminating disjunctions is also $s \cdot t$. Hence the flattened formula may now contain $q^{O(|\varphi|)}$ variables.

In Step 2 of the second round, Gröbner basis computation takes time and space exponential in the number of variables. Therefore, Step 2 can now take $q^{q^{O(|\varphi|)}}$ in time and space.

In Step 3 of the second round, however, the number of conjuncts s *does not* become doubly exponential. This is because g_i in Step 3 no longer contains the exponentially many introduced variables – they were already eliminated in the previous step. Thus s is reduced back to single exponential in the number of the remaining variables; i.e., it is bounded by $q^{O(|\varphi|)}$. Similarly, the Gröbner basis computation on each $g_i u' - 1$, which now contains variables $\boldsymbol{x}_1, ..., \boldsymbol{x}_{m-1}, \boldsymbol{y}$, takes time and space $q^{O(|\varphi|)}$. In all, Step 3 takes time and space $q^{O(|\varphi|)}$.

In sum, the second round of the main loop can take time/space $q^{q^{O(|\varphi|)}}$. But at the end of the loop, the size of formula is reduced to $q^{O(|\varphi|)}$ after the Gröbner basis computations, because it is at most single exponential in the number of the remaining variables. Therefore, the double exponential bound remains for future iterations of the main loop. $\square$

Recently, [18] reports a Gröbner basis computation algorithm in finite fields using polynomial space. This algorithm is theoretical and can not be applied yet. Given the analysis above, if such a polynomial-space algorithm for Gröbner basis computation can be practically used, the intermediate expressions do not have the double-exponential blow-up. On the other hand, it does not lower the space bound of our algorithm to polynomial space, because during flattening of the disjunctions, the introduced terms are multiplied together. To expand the introduced terms, one may still use exponential space. It remains further work to investigate whether this new algorithm can be practically used and how it compares with Buchberger's Algorithm.

Proposition 5.2. *If there exists a polynomial-space Gröbner basis computation algorithm over finite fields for ideals containing the field polynomials, the time/space complexity of our algorithm is bounded by* $q^{O(|\varphi|)}$.

6 Examples

6.1 A Walk-through Example

Suppose we have the following formula over F_3:

$$\varphi : \exists b \forall a \exists y \exists x.((y = ax^2 + bx + c) \wedge (y = ax)),$$

which has three alternating quantifier blocks and one free variable. We ask for a quantifier-free formula $\psi(c)$ equivalent to φ.

We fix the lexicographic ordering to be $x \succ y \succ a \succ b \succ c$. First, we compute the Gröbner basis G_0 of the ideal: $\langle y - ax^2 - bx - c, y - ax, x^3 - x, y^3 - y, a^3 - a, b^3 - b, c^3 - c\rangle$,and obtain the Gröbner basis of the elimination ideal

$$G_1 = G_0 \cap F_3[a, b, c] = \{abc + ac^2 + b^2c - c, a^3 - a, b^3 - b, c^3 - c\}.$$

After this, x and y have been eliminated, and we have:

$$\begin{aligned} [\![\varphi]\!] &= [\![\exists b \forall a.((abc + ac^2 + b^2c - c = 0) \wedge (a^3 - a = 0) \wedge (b^3 - b = 0) \wedge (c^3 - c = 0))]\!] \\ &= [\![\exists b \forall a.(abc + ac^2 + b^2c - c = 0)]\!] \quad \text{(using Lemma 3.3)} \\ &= [\![\exists b.(\neg \exists a \exists u.(u(abc + ac^2 + b^2c - c) - 1 = 0))]\!] \end{aligned}$$

Now we eliminate quantifiers in $\exists a \exists u((abc + ac^2 + b^2c - c) \cdot u - 1 = 0)$, again by computing the Gröbner basis G_2 of the ideal

$$\langle (abc + ac^2 + b^2c - c)u - 1, a^3 - a, b^3 - b, c^3 - c, u^3 - u\rangle \cap F_3[b, c].$$

We obtain $G_2 = \{b^2 - bc, c^2 - 1\}$. Therefore $[\![\varphi]\!] = [\![\exists b(\neg(b^2 - bc = 0 \wedge c^2 - 1 = 0))]\!]$. (Note that if both b and c are both free variables, $b^2 - bc \neq 0 \vee c^2 - 1 \neq 0$ would be the quantifier-free formula containing b, c that is equivalent to φ.)

Next, we introduce u_1 and u_2 to eliminate the negations, and v to eliminate the disjunction: $[\![\varphi]\!] = [\![\exists b \exists u_1 \exists u_2 \exists v.(((b^2 - bc)u_1 - 1)v = 0 \bigwedge ((c^2 - 1)u_2)(1 - v) = 0)]\!]$.

We now do a final step of computation of the Gröbner basis G_3 of:

$$\langle ((b^2 - bc)u_1 - 1)v, ((c^2 - 1)u_2)(1 - v), b^3 - b, c^3 - c, u_1^3 - t_1, u_2^3 - t_2, v^3 - v\rangle \cap F_3[c].$$

We obtain $G_3 = \{c^3 - c\}$. This gives us the result formula $[\![\varphi]\!] = [\![c^3 - c = 0]\!]$, which means that c can take any value in F_3 to make the formula true.

6.2 Analyzing a Biological Controller Design

We studied a virus competition model named S2VD [9], which models the dynamics of virus competition as a polynomial system over finite fields. The authors aimed to design a controller to ensure that one virus prevail in the environment. They pointed out that there was no existing method for verifying its correctness. The current design is confirmed effective by computer simulation and lab experiments for a wide range of initializations. We attempted to establish the correctness of the design with formal verification techniques. However, we found bugs in the design.

All the Gröbner basis computations in this section are done using scripts in the SAGE system [1], which uses the underlying Singular implementation [2]. All the formulas below are solved within 5 seconds on a Linux machine with 2GHz CPU and 2GB RAM. They involve around 20 variables over F_4, with nonlinear polynomials containing multiplicative products of up to 50 terms.

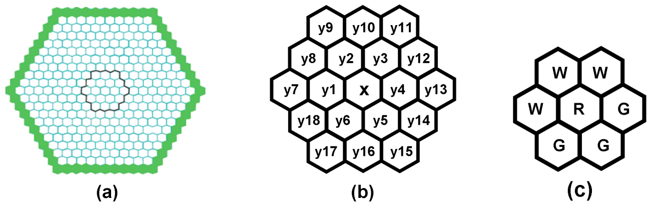

Fig. 1: (a) The ten rings of S2VD; (b) Cell x and its neighbor $\boldsymbol{y}$ cells; (c) The counterexample

The S2VD Model The model consists of a hexagonal grid of cells. Each hexagon represents a cell, and each cell has six neighbors. There are four possible colors for each cell. A green cell is infected with (the good) Virus G, and a red cell is infected with (the bad) Virus R. When the two viruses meet in one cell, Virus G captures Virus R and the cell becomes yellow. A cell not infected by any virus is white. The dynamics of the system is determined by the interaction of the viruses.

There are ten rings of cells in the model, with a total of 331 cells (Figure 1(a)). In the initial configuration, the cells in Ring 4 to 10 are set to white, and the cells in Ring 1 to 3 can start with arbitrary colors. The aim is to have a controller that satisfies the following safety property: The cells in the outermost ring are either green or white at all times. The proposed controller detects if any cell has been infected by Virus R, and injects cells that are "one or two rings away" from it with Virus G. The injected Virus G is used to block the further expansion of Virus R.

Formally, the model is a polynomial system over the finite field $F_4 = \{0, 1, a, a+1\}$, with each element representing one color: $(0, green)$, $(1, red)$, $(a, white)$, $(a+1, yellow)$. The dynamics is given by the function $f : F_4^{331} \to F_4^{331}$. For each cell x, its dynamics f_x is determined by the color of its six neighbors $y_1, ..., y_6$, specified by the nonlinear polynomial $f_x =_{df} \gamma_2^2 + \gamma_2\gamma_1^3 + a^2(\gamma_1^3 + \gamma_1^2 + \gamma_1)$, where $\gamma_1 = \sum_{i=1}^{6} y_i$ and $\gamma_2 = \sum_{i \neq j} y_i y_j$. The designed controller is specified by another function $g : F_4^{331} \to F_4^{331}$: For each cell x, with $y_1, ..., y_{18}$ representing the cells in the two rings surrounding it, we define $g_x =_{df} \prod_{i=1}^{18}(1 - y_i)^3$. More details can be found in [9].

Applying Quantifier Elimination We first try checking whether the safety property itself forms an inductive invariant of the system (which is a strong sufficient check). To this end, we check whether the controlled dynamics of the system remain inside the invariant on the boundary (Ring 10) of the system. Let x be a cell in Ring 10 and $\boldsymbol{y} = (y_1, ..., y_{18})$ be the cells in its immediate two rings. We assume the cells outside Ring 10 $(y_8, ..., y_{12}, y_2, y_3)$ are white. See Figure 1(b) for the coding of the cells. We need to decide the formula:

$$\underbrace{\forall x((\exists \boldsymbol{y}((\bigwedge_{i=8}^{12}(y_i = a) \wedge y_2 = a \wedge y_3 = a) \wedge \text{Safe}(\boldsymbol{y}) \wedge x = F_x(\boldsymbol{y})))}_{\varphi_1} \to \underbrace{x(x-a) = 0)}_{\text{"x stays green/white"}} \quad (1)$$

where (writing $\gamma_1 = \sum_{i=1}^{6} y_i, \gamma_2 = \sum_{i \neq j \in \{1,...,6\}} y_i y_j$)

$$\text{Safe}(\boldsymbol{y}) =_{df} (y_1(y_1 - a) = 0 \wedge y_4(y_4 - a) = 0 \wedge y_7(y_7 - a) = 0 \wedge y_{13}(y_{13} - a) = 0)$$

$$F_x(\boldsymbol{y}) =_{df} (\gamma_2^2 + \gamma_2\gamma_1^3 + a^2(\gamma_1^3 + \gamma_1^2 + \gamma_1)) \cdot (\prod_{i=1}^{18}(1 - y_i))^3$$

After quantifier elimination, Formula (1) turns out to be false. In fact, we obtained $[\![\varphi_1]\!] = [\![x^4 - x = 0]\!]$. Therefore, the safety property itself is not an inductive invariant of the system. We realized that there is an easy counterexample of safety of the proposed controller design: Since the controller is only effective when red cells occur, it does not prevent the yellow cells to expand in all the cells. Although this is already a bug of the system, it may not conflict with the authors' original goal of controlling the red cells. However, a more serious bug is found by solving the following formula:

$$\forall x(\underbrace{(\exists \boldsymbol{y}(\bigwedge_{i=1}^{18} y_i(y_i - a)(y_i - a^2) = 0) \wedge x = F_x(\boldsymbol{y}))}_{\varphi_2} \rightarrow \underbrace{\neg(x = 1)}_{\text{“x is not red”}}) \tag{2}$$

Formula (2) expresses the desirable property that when none of the neighbor cells of x is red, x never becomes red. However, we found again that $[\![\varphi_2]\!] = [\![x^4 - x]\!]$, which means in this scenario the x cell can still turn red. Thus, the formal model is inconsistent with the informal specification of the system, which says that non-red cells can never interact to generate red cells. In fact, the authors mentioned that the dynamics F_x is not verified because of the combinatorial explosion. Finally, to give a counterexample of the design, we solve the formula

$$\varphi_3 =_{df} \exists \boldsymbol{y} \exists x.(x = 1 \wedge \bigwedge_{i=1}^{6} y_i(y_i - a)(y_i - a^2) = 0 \wedge x = F_x(\boldsymbol{y})) \tag{3}$$

The formula checks whether there exists a configuration of $y_1, ..., y_6$ which are all non-red, such that x becomes red. φ_3 evaluates to true. Further, we obtain $x = 1, \boldsymbol{y} = (a, a, a, 0, 0, 0)$ as a witness assignment for φ_3. This serves as the counterexample (see Figure 1(c)).

This example shows how our quantifier elimination procedure provides a practical way of verifying and debugging systems over finite fields that were previously not amenable to existing formal methods and can not be approached by exhaustive enumeration.

7 Conclusion

In this paper, we gave a quantifier elimination algorithm for the first-order theory over finite fields based on the Nullstellensatz over finite fields and Gröbner basis computation. We exploited special properties of finite fields and showed the correspondence between elimination of quantifiers, projection of varieties, and computing elimination ideals. We also generalized the Tseitin transformation from Boolean formulas to formulas over finite fields using ideal operations. The complexity of our algorithm depends on the complexity of Gröbner basis computation. In an application of the algorithm, we successfully found bugs in a biological controller design, where the original authors expressed that no verification methods were able to handle the system. In future work, we expect to use the algorithm to formally verify more systems with finite field arithmetic. Further (theoretical and practical) optimizations on Gröbner basis computation over finite fields are highly desirable.

References

1. The SAGE computer algebra system. `http://sagemath.org`.

2. The Singular computer algebra system. `http://www.singular.uni-kl.de/`.
3. T. Becker and V. Weispfenning. *Gröbner Bases.* Springer, 1998.
4. M. L. Borgne, A. Benveniste, and P. L. Guernic. Polynomial dynamical systems over finite fields. In *Algebraic Computing in Control*, volume 165. Springer.
5. B. Buchberger. A theoretical basis for the reduction of polynomials to canonical forms. *ACM SIGSAM Bulletin*, 10(3):19–29, 1976.
6. D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms.* Springer, 1997.
7. D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry.* Springer, 2005.
8. R. Germundsson. Basic results on ideals and varieties in finite fields. *Technical Report LiTH-ISY-I-1259, Linkoping University, S-581 83*, 1991.
9. A. Jarrah, H. Vastani, K. Duca, and R. Laubenbacher. An optimal control problem for in vitro virus competition. In *43rd IEEE Conference on Decision and Control*, 2004.
10. A. S. Jarrah, R. Laubenbacher, B. Stigler, and M. Stillman. Reverse-engineering of polynomial dynamical systems. In *Advances in Applied Mathematics*, volume 39, pages 477–489, 2007.
11. Y. N. Lakshman. On the complexity of computing a Gröbner basis for the radical of a zero dimensional ideal. In *STOC '90*, pages 555–563, New York, NY, USA, 1990. ACM.
12. S. Lang. *Algebra.* Springer, 3rd edition, 2005.
13. H. Marchand and M. L. Borgne. On the optimal control of polynomial dynamical systems over Z/pZ. In *In 4th International Workshop on Discrete Event Systems*, pages 385–390, 1998.
14. D. Marker. Model theory. Springer, 2002.
15. D. Monniaux. A quantifier elimination algorithm for linear real arithmetic. In *LPAR*, 2008.
16. T. Nipkow. Linear quantifier elimination. In *IJCAR*, pages 18–33, 2008.
17. E. W. Smith and D. L. Dill. Automatic formal verification of block cipher implementations. In *FMCAD*, pages 1–7, 2008.
18. Q.-N. Tran. Groebner bases computation in boolean rings is P-SPACE. In *International Journal of Applied Mathematics and Computer Sciences*, volume 5, No. 2, 2009.

Appendix A: Omitted Proofs

Proof of Lemma 2.1 This is a consequence of the Seidenberg's Lemma (Lemma 8.13 in [3]). It can also be directly proved as follows.

Proof. We need to show $\sqrt{J + \langle x_1^q - x_1, ..., x_n^q - x_n \rangle} = J + \langle x_1^q - x_1, ..., x_n^q - x_n \rangle$. Since by definition, any ideal is contained in its radical, we only need to prove

$$\sqrt{J + \langle x_1^q - x_1, ..., x_n^q - x_n \rangle} \subseteq J + \langle x_1^q - x_1, ..., x_n^q - x_n \rangle.$$

Consider an arbitrary polynomial $f \in \sqrt{J + \langle x_1^q - x_1, ..., x_n^q - x_n \rangle}$. By definition, for some integer s, $f^s \in J + \langle x_1^q - x_1, ..., x_n^q - x_n \rangle$. Let $[f]$ and $[J]$ be the images of, respectively, f and J, in $R/\langle x_1^q - x_1, ..., x_n^q - x_n \rangle$ under the canonical homomorphism from R to $R/\langle x_1^q - x_1, ..., x_n^q - x_n \rangle$. For brevity we write $S = \langle x_1^q - x_1, ..., x_n^q - x_n \rangle$.

Now we have $[f]^s \in [J]$, and we further need $[f] \in [J]$. We prove, by induction on the structure of polynomials, that for any $[g] \in R/S$, $[g]^q = [g]$.

– If $[g] = cx_1^{a_1} \cdots x_n^{a_n} + S$ $(c \in F_q, a_i \in N)$, then

$$[g]^q = (cx_1^{a_1} \cdots x_n^{a_n} + S)^q = (cx_1^{a_1} \cdots x_n^{a_n})^q + S = cx_1^{a_1} \cdots x_n^{a_n} + S = [g].$$

– If $[g] = [h_1] + [h_2]$, by inductive hypothesis, $[h_1]^q = [h_1], [h_2]^q = [h_2]$, and, since any element divisible by p is zero in F_q $(q = p^r)$, then

$$[g]^q = ([h_1] + [h_2])^q = \sum_{i=0}^{q} \binom{q}{i} [h_1]^i [h_2]^{q-i} = [h_1]^q + [h_2]^q = [h_1] + [h_2] = [g]$$

Hence $[g]^q = [g]$ for any $[g] \in R/S$, without loss of generality we can assume $s < q$ in $[f]^s$. Then, since $[f]^s \in [J]$, $[f] = [f]^q = [f]^s \cdot [f]^{q-s} \in [J]$. □

Proof of Theorem 2.2

Proof. For an arbitrary ideal $J \subseteq F_q[x_1, ..., x_n]$, applying Hilbert's Nullstellensatz to $J + \langle x_1^q - x_1, ..., x_n^q - x_n \rangle$ and using the previous lemma, we have:

$$I(V^a(J + \langle x_1^q - x_1, ..., x_n^q - x_n \rangle)) = J + \langle x_1^q - x_1, ..., x_n^q - x_n \rangle$$

But since $V^a(\langle x_1^q - x_1, ..., x_n^q - x_n \rangle) = F_q^n$,

$$V^a(J + \langle x_1^q - x_1, ..., x_n^q - x_n \rangle) = V^a(J) \cap F_q^n = V(J)$$

and we reach the clean form, $I(V(J)) = J + \langle x_1^q - x_1, ..., x_n^q - x_n \rangle$. □

Proof of Lemma 3.1

Proof. Let $\boldsymbol{a} \in F_q^{n+m}$ be an assignment vector for $(\boldsymbol{x}, \boldsymbol{y})$.

If $\boldsymbol{a} \in [\![\bigwedge_{i=1}^r f_i = 0]\!]$, then $f_1(\boldsymbol{a}) = \cdots = f_r(\boldsymbol{a}) = 0$ and $\boldsymbol{a} \in V(\langle f_1, ..., f_k \rangle)$.

If $\boldsymbol{a} \in V(\langle f_1, ..., f_r \rangle)$, then $\bigwedge_{i=1}^r f_i(\boldsymbol{a}) = 0$ is true and $\boldsymbol{a} \in [\![\bigwedge_{i=1}^r f_i = 0]\!]$. □

Proof of Lemma 3.2

Proof. We show set inclusion in both directions.

– For any $\boldsymbol{b} \in [\![\exists \boldsymbol{x} \varphi(\boldsymbol{x}; \boldsymbol{y})]\!]$, by definition, there exists $\boldsymbol{a} \in F_q^n$ such that $(\boldsymbol{a}, \boldsymbol{b})$ satisfies $\varphi(\boldsymbol{x}; \boldsymbol{y})$. Therefore, $(\boldsymbol{a}, \boldsymbol{b}) \in [\![\varphi(\boldsymbol{x}; \boldsymbol{y})]\!]$, and $\boldsymbol{b} \in \pi_n([\![\varphi(\boldsymbol{x})]\!])$.
– For any $\boldsymbol{b} \in \pi_n([\![\varphi(\boldsymbol{x}; \boldsymbol{y})]\!])$, there exists $\boldsymbol{a} \in F_q^n$ such that $(\boldsymbol{a}, \boldsymbol{b}) \in [\![\varphi(\boldsymbol{x})]\!]$. By definition, $\boldsymbol{b} \in [\![\exists x \varphi(\boldsymbol{x}; \boldsymbol{y})]\!]$. □

Proof of Lemma 3.3

Proof. We have $[\![\bigwedge_{i \in A_x}(x_i^q - x_i = 0) \wedge \bigwedge_{i \in A_y}(y_i^q - y_i = 0)]\!] = [\![\top]\!]$, which follows from Proposition 2.1. □

Proof of Lemma 4.1

Proof. Let $\varphi[\psi_1/\psi_2]$ denote substitution of ψ_1 in φ by ψ_2. Suppose the negative atomic formulas in φ are $f_1 \neq 0, ..., f_k \neq 0$.

We introduce a new variable z_1, and substitute $f_1 \neq 0$ by $p \cdot z_1 = 1$. Since the field F_q does not have zero divisors, all the solutions for $[\![p \neq 0]\!] = [\![\exists z_1(p \cdot z_1 = 1)]\!]$ (the Rabinowitsch trick).

Iterating the procedure, we can use k new variables $z_1, ..., z_k$ so that:

$$[\![\varphi]\!] = [\![\varphi[f_1 \neq 0/(\exists z_1.(p \cdot z_1 - 1 = 0))] \cdots [f_k \neq 0/(\exists z_k.(p \cdot z_k - 1 = 0))]]\!]$$

Since the result formula contains no more negations and the z_is are new variables, it can be put into prenex form $\exists \boldsymbol{z}.(\varphi[f_1 \neq 0/(p \cdot z_1 - 1 = 0)] \cdots [f_k \neq 0/(p \cdot z_k - 1 = 0)])$. □

Proof of Lemma 4.2

Proof. $[\![\psi_1 \vee \psi_2]\!] = V(J_1) \cup V(J_2)$ follows from the definition of realization. We only need to show the second equality. Let $\boldsymbol{a} = (a_1, ..., a_n) \in F_q^n$ be a point.

- Suppose $\boldsymbol{a} \in V(J_1) \cup V(J_2)$. If $\boldsymbol{a} \in V(J_1)$, then $(1, a_1, ..., a_n) \in V(x_0 J_1 + (1 - x_0) J_2)$. If $\boldsymbol{a} \in V(J_2)$, then $\langle 0, a_1, ..., a_n \rangle \in V(x_0 J_1 + (1 - x_0) J_2)$. In both cases, $\boldsymbol{a} \in \pi_0(V(x_0 J_1 + (1 - x_0) J_2))$.

- Suppose $\boldsymbol{a} \in \pi_0(V(x_0 J_1 + (1 - x_0) J_2))$. There exists $a_0 \in F_q$ such that $(a_0, a_1, ..., a_n) \in V(x_0 J_1 + (1 - x_0) J_2)$. If $a_0 \notin \{0, 1\}$, then all the polynomials in J_1 and J_2 need to vanish on $\boldsymbol{a}$; if $a_0 = 1$ then J_1 vanishes on $\boldsymbol{a}$; if $a_0 = 0$ then J_2 vanishes on $\boldsymbol{a}$. In all cases, $\boldsymbol{a} \in V(J_1) \cup V(J_2)$. □

Proof of Theorem 5.1

Proof. We only need to show the intermediate formulas obtained in Step 1-3 are always equivalent to the original formula φ. In Step 1, the formula is flattened with ideal operations, which preserve the realization of the formula as proved in

Theorem 4.1. In Step 2, we have (by Theorem 3.1) $[\![\exists \boldsymbol{x}_m \exists \boldsymbol{t} \exists \boldsymbol{s} (\bigwedge_{i=1}^{r} (f_i = 0))]\!] = [\![\bigwedge_{i=1}^{u} (g_i = 0)]\!]$.

Hence the formula obtained in Step 2 is equivalent to φ. In Step 3, the substitution preserves realization of the formula because

$$[\![\bigwedge_{i=1}^{u} \forall \boldsymbol{x}_{m-1} (g_i = 0)]\!] = [\![\bigwedge_{i=1}^{u} (\neg \exists \boldsymbol{x}_{m-1} (\neg g_i = 0))]\!] = [\![(\bigwedge_{i=1}^{u} (\bigvee_{j=1}^{v_i} h_{ij} \neq 0))]\!],$$

where the second equality is guaranteed by Theorem 3.1 again.

The loop terminates either at the end of Step 2 or Step 3. Hence the output quantifier-free formula ψ is always in conjunctive normal form, which contains only variables $\boldsymbol{y}$, and is equivalent to the original formula φ. $\square$

Appendix B: Omitted Algorithms

Algorithm 2 Eliminate Negations

```
Input: $\psi$, which is quantifier-free and in negation normal form.
Output: $(\psi, \boldsymbol{t})$, where $\psi$ contains no negations and $[\![\exists \boldsymbol{t}.\psi]\!] = [\![\psi]\!]$.
Procedure Eliminate_Negations($\psi$){
$\{f_1 \neq 0, \cdots f_r \neq 0\} \leftarrow$ Negative atomic formulas in $\varphi$
for $i = 1$ to $r$ do
    $\psi \leftarrow \psi[(p \neq 0)/(p \cdot t_i - 1 = 0)]$
    $\boldsymbol{t} \leftarrow \boldsymbol{t} \cup t_i$
end for
return $(\psi, \boldsymbol{t})$}
```

Algorithm 3 Formula to Ideals

```
1: Input: A quantifier-free $\psi$, which is in negation normal form but contains
   no negations.
2: Output: $(J, \boldsymbol{z})$, where $J$ is an ideal and $\boldsymbol{z}$ are new variables.
3: Procedure Formula_to_Ideal($\psi$){
4: if $\psi == (f = 0)$ then
5:     $J \leftarrow \langle f \rangle$
6: end if
7: if $\psi == \psi_1 \wedge \psi_2$ then
8:     $J \leftarrow$ Formula_to_Ideal($\psi_1$)+Formula_to_Ideal($\psi_2$)
9: end if
10: if $\psi == \psi_1 \vee \psi_2$ then
11:     $J \leftarrow$ Formula_to_Ideal($\psi_1$)$\cdot z_i$+Formula_to_Ideal($\psi_2$)$(1 - z_i)$
12:     $\boldsymbol{z} \leftarrow \boldsymbol{z} \cup \{z_i\}$
13: end if
14: return $(J, \boldsymbol{z})$ }
15:
16: Procedure Formula_Flattening($\psi$){
17: $(J, \boldsymbol{z}) \leftarrow$ Formula_to_Ideal($\psi$)
18: $\langle f_1, ..., f_r \rangle \leftarrow J$
19: return $(\bigwedge_{i=1}^{r}(f_i = 0), \boldsymbol{z})$}
```

Appendix D: Complexity of Gröbner Basis Computation over Finite Fields

[Proposition 6.1] *For ideals of the form* $\langle f_1, ..., f_m, x_1^q - x_1, ..., x_n^q - x_n \rangle \subseteq F_q[x_1, ..., x_n]$*, where* $f_1, ..., f_m$ *are given in expanded form and have longest length of* l*, the basic Buchberger Algorithm halts in time* $q^{O(n)} + O(m^2 l)$*.*

Proof. At the beginning, we need to calculate the steps needed for preprocessing (Line 4). Preprocessing is not always necessary, and the reason for doing this is to avoid having some f_i with degree higher than q on some variable, as well as duplicated leading monomials in $LM(f_1), ..., LM(f_m)$. First, we divide each f_i by the field polynomials, which can be easily done by replacing each occurrence of x_i^q by x_i, if there are any. This requires at most $O(ml)$ steps. Then we need to make sure all the duplicated leading monomials in $f_1, ..., f_m$ are deleted, as follows. After ordering the polynomials with respect to their leading monomials and $<_M$, we search for duplicated leading monomials. Suppose f_i and f_j have the same leading monomial, then we let f_j' to be the result of subtracting $(LT(f_j)/LT(f_i))f_i$ from f_j, so that the leading monomial of f_j is canceled. Now f_j' is a new polynomial whose leading monomial is strictly smaller than that of f_j. We delete f_j and put f_j' back into the polynomial list. If f_j' still has duplicated leading monomial with other polynomials down the list, we repeat the subtraction and inserting operations. Completing this process requires $O(ml)$ at

Algorithm 4 Buchberger's Algorithm

```
1: INPUT: $f_1, ..., f_m, x_1^q - x_1, ..., x_n^q - x_n \in F_q[x_1, ..., x_n]$ and a monomial ordering $<_M$.
2: OUTPUT: $N$ - the number of satisfying assignments of $\varphi$.
3: $G \leftarrow \{f_1 - 1, ..., f_m - 1, x_1^q - x_1, ..., x_n^q - x_n\}$
4: Preprocess G so that $f_1, ..., f_m$ are reduced with respect to $x_i^q - x_i$ and have no duplicated leading monomials.
5: while $G \neq \emptyset$ do
6:     $S \leftarrow \emptyset$
7:     Order the polynomials in G as $(g_1, ..., g_t)$ in lexicographic order
8:     for $1 \leq i < j \leq t$ do
9:         $h \leftarrow \frac{LCM(LT(g_i),LT(g_j))}{LT(g_j)} g_i - \frac{LCM(LT(g_i),LT(g_j))}{LT(g_i)} g_j$
10:        $r \leftarrow$ the remainder after reducing $h$ with respect to $g_1, ..., g_t$
11:        if $r \neq 0$ then
12:            $S \leftarrow S \cup \{r\}$
13:            $G \leftarrow G \cup S$
14:        end if
15:    end for
16: end while
```

most for each polynomial, and at most $O(m^2 l)$ for all the polynomials. In all, completing preprocessing requires $T_{preprocessing} = O(m^2 l)$.

Now we analyze the main loop (Line 5 to 16). Consider round $k+1$ of the main loop. Let T_{k+1} be the time taken by this round and L_{k+1} the size of G after this round. We immediately have the following relation between T_{k+1} and L_k:

$$T_{k+1} \leq \binom{L_k}{2} \cdot T_{Spoly} \cdot T_{reduce},$$

where T_{Spoly} is the time taken by computing the S-polynomial from a pair of polynomials in G (Line 9) and T_{reduce} is the maximum time taken by reducing an S-polynomial with respect to polynomials in G (Line 10).

The standard reduction process uses multivariate polynomial division [6]. Since during the division process, after each division round, the leading monomial of the dividend polynomial strictly decreases with respect to the monomial ordering, the number of division rounds is bounded by the longest downward monomial chain leading from $LM(h)$ (h as defined on Line 9) that can appear in the division. Now, the existence of the field polynomials $x_i^q - x_i$ in the divisor pool ensures two properties of the division process (Line 8):

First, after division, any variable appearing in the remainder must have degrees lower than q, otherwise the division should not halt. Hence the longest possible remainder is shorter than q^n, which is the maximum number of different monomials with degree on each variable lower than q. Notice that we have preprocessed the input polynomials to have no duplicated leading monomials, and the multivariate division procedure ensures that the remainder can only have a strictly smaller leading monomial compared to the dividend, which is different from all the leading monomials of the divisors. Hence, after each round the length of the basis (the number of polynomials in G) is always smaller than $q^n + n$, since we can at most have q^n different monomials with degree on each variable smaller than q, plus n extra field polynomials with degree q.

Second, when an S-polynomial or intermediate polynomial appearing during the division as dividend has a degree higher than q on variable x_i, the degree is reduced to lower than q, through division by $x_i^q - x_i$. Since the length of any downward chain from the leading monomial of the dividend is at most q^n, and at each division round the reduction by $x_i^q - x_i$ may take up another step, we have the maximum number of division rounds bounded by $2q^n$.

Further, notice that when operating over a finite field, the positive characteristic ensures that the coefficients never grow exponentially as in the case for rationals or complex numbers. We can assume arithmetic among coefficients takes unit time. Each division round, say dividing f by g, consists of multiplying $LT(f)/LT(g)$ to g, and subtracting $(LT(f)/LT(g)) \cdot g$ from f. Thus each round during the division process takes time at most $2q^n$, since q^n is the longest possible length of f and g.

Finally, S-polynomials are defined as (cf. [6]):

$$S(g_i, g_j) = \frac{LCM(LT(g_i), LT(g_j))}{LT(g_i)} \cdot g_i - \frac{LCM(LT(g_i), LT(g_j))}{LT(g_j)} \cdot g_j$$

Hence the number of arithmetic steps taken by computing S-polynomial, $T_{S-polynomial}$, is less than the sum of lengths of the two polynomials, which is also bounded by $2q^n$.

In all, we have $T_{reduce} \leq 2q^n \cdot 2q^n$, $L_k \leq q^n + n$ and $T_{Spoly} \leq 2q^n$. Hence

$$T_{k+1} \leq \binom{L_k}{2} \cdot T_{Spoly} \cdot T_{reduce} \leq (q^n + n)^2 \cdot 2q^n \cdot 2q^n \cdot 2q^n$$

Now observe that the final length of the Gröbner basis is bounded by $q^n + n$ (again, because we can have at most q^n different leading monomials appearing in G, and n extra field polynomials). At each round at least one polynomial with a new leading monomial needs to be generated, otherwise the process halts. Hence the maximum number of rounds is bounded by $(q^n + n) - (m + n)$. (We preprocessed the input polynomials $f_1, ..., f_m$ to be mutually irreducible and already reduced by $x_i^q - x_i$, hence $m < q^n$.)

In all, the overall worst-case complexity of Buchberger's Algorithm is

$$\begin{aligned} T &\leq (q^n + n - (m + n)) \cdot T_k + T_{preprocessing} \\ &\leq q^n \cdot (q^n + n)^2 \cdot 2q^n \cdot 2q^n \cdot 2q^n + O(m^2 l) = q^{O(n)} + O(m^2 l). \end{aligned}$$

At last, since there are only q^n possible different standard monomials, the worst-case complexity of the counting step is $O(q^n)$, subsumed by the complexity of Buchberger's Algorithm. We'll discuss practical heuristics for the counting step in the next section. Here we settle for the overall worst-case complexity, which is $q^{O(n)} + O(m^2 l)$. □